

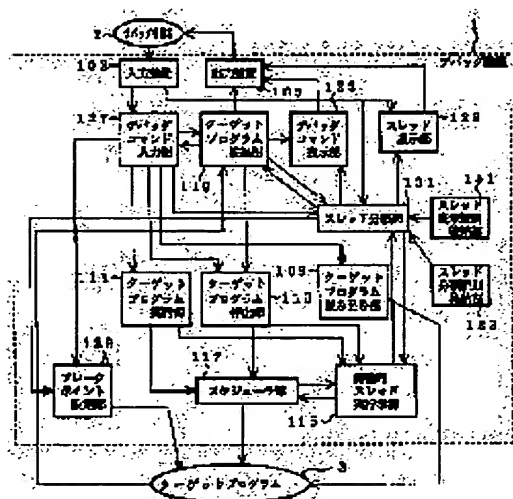
(11)Publication number : 11-338733
(43)Date of publication of application : 10.12.1999

G06F 11/28
G06F 11/28

(72)Inventor : TAMURA FUMITAKA

Priority number : 10 48424 Priority date : 27.02.1998 Priority country : JP

SOLUTION: Concerning a debugging device 1 with a multi-thread program as an object, a thread classifying part 121 classifies threads into exclusive execution threads and the other threads and as debugging commands, commands individually able to be applied to the threads and the other commands are provided. Then, control is performed so that the threads designated as exclusive execution threads can not be simultaneously executed. Thus, the plural threads can be executed in the desired order of a debugging worker. Therefore, the debugging worker can reproduce bugging caused by timing at all the time.



<http://www19.ipdl.jpo.go.jp/PA1/result/detail/main/wAAAEWaOePDA411338733P...> 2003/10/10

[Date of requesting appeal against examiner's
decision of rejection]

[Date of extinction of right]

Copyright (C); 1998,2003 Japan Patent Office

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平11-338733

(43) 公開日 平成11年(1999)12月10日

(51) Int.Cl.⁶

G 0 6 F 11/28

識別記号

3 1 0

F I

G 0 6 F 11/28

3 1 0 B

A

審査請求 未請求 請求項の数48 OL (全 36 頁)

(21) 出願番号 特願平11-53392

(22) 出願日 平成11年(1999)3月1日

(31) 優先権主張番号 特願平10-48424

(32) 優先日 平10(1998)2月27日

(33) 優先権主張国 日本 (J P)

(71) 出願人 000003078

株式会社東芝

神奈川県川崎市幸区堀川町72番地

(72) 発明者 田村 文隆

神奈川県川崎市幸区柳町70番地 株式会社

東芝柳町工場内

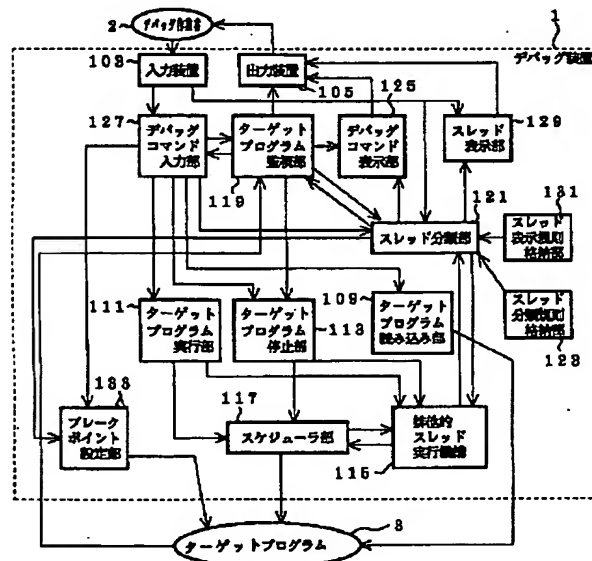
(74) 代理人 弁理士 須山 佐一

(54) 【発明の名称】 デバッグ装置及び記録媒体

(57) 【要約】

【課題】 プログラム中の誤りを効率良くデバッグすることができるデバッグ装置及び記録媒体を提供すること。

【解決手段】 マルチスレッドプログラムを対象とするデバッグ装置1において、スレッド分類部121がスレッドを排他実行スレッドとそうでないスレッドに分類し、デバッグコマンドとしてスレッドに個別に与え得るものとプログラム全体に与え得るものとを設け、排他実行スレッドに指定されたものは同時に実行させないように制御することにより、複数のスレッドの実行を、デバッグ作業者の希望する順序で実行することが可能となり、従ってデバッグ作業者は、タイミングに起因するバグの再現を常に行うことができるようになる。



【特許請求の範囲】

【請求項 1】 並列して実行される逐次実行単位を含むプログラムについてのデバッグの制御を行う制御手段と、

前記逐次実行単位の内、特定の逐次実行単位のみについて前記制御手段を動作させる対象特定制御手段とを具備することを特徴とするデバッグ装置。

【請求項 2】 前記対象特定制御手段が、デバッグ制御の対象となる特定の逐次実行単位のみについて実行の再開または停止を行わせる手段をさらに具備することを特徴とする請求項 1 記載のデバッグ装置。

【請求項 3】 前記対象特定制御手段は、前記各逐次実行単位の分類を指定するための手段と、前記指定された分類を表す分類情報を入力するための手段と、前記入力された分類情報に従ってデバッグ制御を行う手段とを具備することを特徴とする請求項 1 または 2 記載のデバッグ装置。

【請求項 4】 前記対象特定制御手段は、前記各逐次実行単位の分類のための規則を記録する手段と、前記記録された規則を用いて前記各逐次実行単位の初期の分類を定める手段と、前記分類を変更するための手段とをさらに具備することを特徴とする請求項 3 記載のデバッグ装置。

【請求項 5】 前記分類情報は、デバッグ装置が前記逐次実行単位を制御する形態に関する情報であることを特徴とする請求項 3 または 4 記載のデバッグ装置。

【請求項 6】 前記分類情報は、前記逐次実行単位についてデバッグ作業者が着目する程度に関する情報であることを特徴とする請求項 3 または 4 記載のデバッグ装置。

【請求項 7】 前記分類情報は、デバッグ装置が前記逐次実行単位の状態を表示させる形態に関する情報であることを特徴とする請求項 3 または 4 記載のデバッグ装置。

【請求項 8】 並列して実行される逐次実行単位を含むプログラムをデバッグするデバッグ装置において、前記デバッグ装置は、前記プログラムの状態を制御する指示を入力するための手段を具備し、

前記指示は、前記プログラムの状態のうち、特定の逐次実行単位に関する実行状態のみを対象とする特定制御指示と、前記プログラムの状態全体を対象とする一般制御指示とを含むことを特徴とするデバッグ装置。

【請求項 9】 デバッグ対象のプログラムの状態全体を対象とする制御を行うための制御指示を入力する一般制御指示入力手段と、

前記デバッグ対象のプログラムに含まれる特定の逐次実

行単位を対象として、該逐次実行単位の実行状態の制御を行うための制御指示を入力する特定制御指示入力手段と、

前記デバッグ対象のプログラムに含まれる各逐次実行単位を分類する分類手段と、

前記一般制御指示入力手段もしくは特定制御指示入力手段にプログラムの状態を制御するための指示が入力された場合には、前記分類手段による分類に基づく制御情報に従ってプログラムの状態の制御を行う制御手段とを具備することを特徴とするデバッグ装置。

【請求項 10】 デバッグ対象のプログラムの状態全体を対象とする制御を行うための制御指示を入力する一般制御指示入力手段と、

前記デバッグ対象のプログラムに含まれる各逐次実行単位を、被制御形態に基づいて排他的逐次実行単位、並立的逐次実行単位及び継続的逐次実行単位のいずれかに分類する分類手段と、

前記一般制御指示入力手段に逐次実行単位の実行を停止する指示が入力された場合には、前記分類手段により排他的逐次実行単位及び並立的逐次実行単位に分類された逐次実行単位のうち実行中の逐次実行単位の実行を停止する停止手段と、

前記一般制御指示入力手段に逐次実行単位の実行を再開する指示が入力された場合には、前記分類手段により排他的逐次実行単位及び並立的逐次実行単位に分類された逐次実行単位のうち停止された逐次実行単位の実行を再開する再開手段とを具備することを特徴とするデバッグ装置。

【請求項 11】 デバッグ対象のプログラムを構成する逐次実行単位のうち特定の逐次実行単位を対象として、該逐次実行単位の実行状態の制御を行うための制御指示を入力する特定制御指示入力手段と、

前記デバッグ対象のプログラムに含まれる各逐次実行単位を、被制御形態に基づいて排他的逐次実行単位、並立的逐次実行単位及び継続的逐次実行単位のいずれかに分類する分類手段と、

前記特定制御指示入力手段に逐次実行単位の実行を停止する指示が入力された場合には、前記分類手段により並立的逐次実行単位に分類された逐次実行単位のうち実行中の総ての逐次実行単位及び前記分類手段により排他的逐次実行単位に分類された逐次実行単位のうち前記特定制御指示入力手段によって第 1 の指定がされた逐次実行単位の実行を停止する停止手段と、

前記特定制御指示入力手段に逐次実行単位の実行を再開する指示が入力された場合には、前記分類手段により並立的逐次実行単位に分類された逐次実行単位の総て、及び排他的逐次実行単位に分類された逐次実行単位のうち前記特定制御指示入力手段によって第 2 の指定がされた逐次実行単位の実行を再開する再開手段とを具備することを特徴とするデバッグ装置。

3

【請求項 1 2】 前記制御手段は、前記デバッグ対象のプログラムに含まれる各逐次実行単位の状態指標に対応させて該逐次実行単位の実行状態の制御を行う手段を具備することを特徴とする請求項 9 記載のデバッグ装置。

【請求項 1 3】 内在的停止命令を含む命令群を逐次実行する特定の逐次実行単位を含むプログラムのデバッグ装置において、前記特定の逐次実行単位に対する単位実行指示を入力するための特定制御指示入力手段と、前記特定制御指示入力手段に第 1 の単位実行指示が入力されたときに前記特定の逐次実行単位が次に実行すべき命令が前記内在的停止命令である場合には前記特定の逐次実行単位を実行停止状態とし、前記特定制御指示入力手段に前記実行停止状態とされた逐次実行単位に対しての第 2 の単位実行指示が入力されると該逐次実行単位が前記内在的停止命令の次の命令を実行可能な状態にする手段とを具備することを特徴とするデバッグ装置。

【請求項 1 4】 命令群を有するモジュールが定義されるデバッグ対象のプログラムに含まれる各モジュールを着目形態において分類する分類手段と、前記分類手段による着目形態に応じたデバッグ制御を行う手段とを具備することを特徴とするデバッグ装置。

【請求項 1 5】 前記分類手段は、前記モジュールを分類のための規則を用いて分類する手段と、外部からの指示を入力するための手段と、前記入力された指示に基づいて前記モジュールの分類を更新する手段とを具備することを特徴とする請求項 1 4 記載のデバッグ装置。

【請求項 1 6】 デバッグ対象のプログラム内のモジュールで定義される命令群に対する単位実行指示が入力される制御指示入力手段と、前記各モジュールを着目属性に応じて着目モジュール及び非着目モジュールに分類する分類手段と、前記制御指示入力手段に前記非着目モジュール内で定義された命令群への単位実行指示が入力された場合には該命令群全体を実行する手段とを具備することを特徴とするデバッグ装置。

【請求項 1 7】 デバッグ対象のプログラムに含まれる並列して実行される逐次実行単位が実行する一、又は、複数の命令群を有するモジュールを分類するモジュール分類手段と、前記各逐次実行単位を、前記モジュール分類手段による分類を考慮して定められる分類規則に基づいてプログラムの実行時に動的に分類する逐次実行単位分類手段と、前記逐次実行単位分類手段による分類情報に従ってデバッグ制御を行う手段とをさらに具備することを特徴とする請求項 1 または 2 記載のデバッグ装置。

【請求項 1 8】 並列して実行される逐次実行単位を含

4

むプログラムの動作を制御するための第 1 の制御指示群を記録する手段と、前記制御指示群に従って前記プログラムの動作を制御することで得られる前記プログラムの第 1 の動作過程を記録する動作過程記録手段と、前記動作過程記録手段に記録された第 1 の動作過程と同じ、又は、異なる動作をするように該各逐次実行単位を制御する第 2 の制御指示群を生成する手段とを具備することを特徴とするデバッグ装置。

10 【請求項 1 9】 前記生成された第 2 の制御指示群に従って前記プログラムの動作を制御する手段と、前記実行によって得られる第 2 の動作過程と前記第 1 の動作過程とを比較する手段とをさらに具備することを特徴とする請求項 1 8 記載のデバッグ装置。

【請求項 2 0】 デバッグ対象のプログラムに含まれる並列して実行される逐次実行単位を属性に応じて分類する分類手段と、前記分類手段により特定の属性を持つと分類される逐次実行単位の動作状態を、該逐次実行単位ごとに設けられたウィンドウに表示する手段と、

20 前記ウィンドウごとに、該ウィンドウに対応する逐次実行単位のみを対象とする第 1 の制御指示を入力するための手段と、前記プログラム全体を対象とする第 2 の制御指示を入力するための手段とを具備することを特徴とするデバッグ装置。

【請求項 2 1】 実行中の逐次実行単位の増減及び前記分類手段による逐次実行単位の分類の変化に対応させて前記ウィンドウをプログラムの実行時に動的に増減させる手段とをさらに具備することを特徴とする請求項 2 0 記載のデバッグ装置。

30 【請求項 2 2】 デバッグ対象のプログラムに含まれる並列して実行される逐次実行単位を分類する手段と、前記逐次実行単位ごとに設けられたウィンドウに該逐次実行単位の動作状態を前記分類に適応させて可視的に表示する手段と、前記プログラム全体を対象とする第 1 の制御指示を入力するための手段と、前記ウィンドウごとに該ウィンドウに対応する逐次実行単位のみを対象とする第 2 の制御指示を入力するための手段とを具備することを特徴とするデバッグ装置。

【請求項 2 3】 デバッグ対象のプログラム内の並列実行される逐次実行単位ごとに設けられたウィンドウ画面内に、前記逐次実行単位に対する前記第 2 の制御指示を入力するための手段のための表示を該逐次実行単位の実行状態に適応させて行う手段を具備することを特徴とするデバッグ装置。

【請求項 2 4】 前記逐次実行単位ごとに設けられたウィンドウ画面ごとに、

50

5

前記逐次実行単位が実行中のモジュールもしくは命令群を表示する手段と、

該ウィンドウに前記モジュールを構成する命令群を呼出す論理軌跡を表示する手段とをさらに具備することを特徴とする請求項 2 記載のデバッグ装置。

【請求項 2 5】 並列して実行される逐次実行単位を含むデバッグ対象プログラムについてのデバッグの制御を行う制御手段と、

前記逐次実行単位のうち特定の逐次実行単位のみについて前記制御手段を動作させる対象特定制御手段とをコンピュータを機能させるためのプログラムとして記録した記録媒体。

【請求項 2 6】 前記対象特定制御手段が、デバッグ制御の対象となる特定の逐次実行単位のみについて実行の再開または停止を行わせる手段をさらに具備することを特徴とする請求項 2 5 記載の記録媒体。

【請求項 2 7】 前記対象特定制御手段は、前記各逐次実行単位の分類を指定するための手段と、前記指定された分類を表す分類情報を入力するための手段と、

前記入力された分類情報に従ってデバッグ制御を行う手段とを具備することを特徴とする請求項 2 5 または 2 6 記載の記録媒体。

【請求項 2 8】 前記対象特定制御手段は、前記各逐次実行単位の分類のための規則を記録する手段と、

前記記録された規則を用いて前記各逐次実行単位の初期の分類を定める手段と、前記分類を変更するための手段とをさらに具備することを特徴とする請求項 2 7 記載の記録媒体。

【請求項 2 9】 前記分類情報は、前記逐次実行単位を制御する形態に関する情報であることを特徴とする請求項 2 7 または 2 8 記載の記録媒体。

【請求項 3 0】 前記分類情報は、前記逐次実行単位についてデバッグ作業者が着目する程度に関する情報であることを特徴とする請求項 2 7 または 2 8 記載の記録媒体。

【請求項 3 1】 前記分類情報は、前記逐次実行単位の状態を表示させる形態に関する情報であることを特徴とする請求項 2 7 または 2 8 記載の記録媒体。

【請求項 3 2】 並列して実行される逐次実行単位を含むデバッグ対象プログラムの状態を制御する指示を入力するための手段がコンピュータを機能させるためのプログラムとして記録され、

前記指示は、前記デバッグ対象プログラムの状態のうち、特定の逐次実行単位に関する実行状態のみを対象とする特定制御指示と、

前記デバッグ対象プログラムの状態全体を対象とする一

6

般制御指示とを含むことを特徴とする記録媒体。

【請求項 3 3】 デバッグ対象プログラムの状態全体を対象とする制御を行うための制御指示を入力する一般制御指示入力手段と、

前記デバッグ対象プログラムに含まれる特定の逐次実行単位を対象として、該逐次実行単位の実行状態の制御を行うための制御指示を入力する特定制御指示入力手段と、

前記デバッグ対象プログラムに含まれる各逐次実行単位を分類する分類手段と、

前記一般制御指示入力手段もしくは特定制御指示入力手段に前記デバッグ対象プログラムの状態を制御するための指示が入力された場合には、前記分類手段による分類に基づく制御情報に従って前記デバッグ対象プログラムの状態の制御を行う制御手段とをコンピュータを機能させるためのプログラムとして記録した記録媒体。

【請求項 3 4】 デバッグ対象のプログラムの状態全体を対象とする制御を行うための制御指示を入力する一般制御指示入力手段と、

前記デバッグ対象プログラムに含まれる各逐次実行単位を、被制御形態に基づいて排他的逐次実行単位、並立的逐次実行単位及び継続的逐次実行単位のいずれかに分類する分類手段と、

前記一般制御指示入力手段に逐次実行単位の実行を停止する指示が入力された場合には、前記分類手段により排他的逐次実行単位及び並立的逐次実行単位に分類された逐次実行単位のうち実行中の逐次実行単位の実行を停止する停止手段と、

前記一般制御指示入力手段に逐次実行単位の実行を再開する指示が入力された場合には、前記分類手段により排他的逐次実行単位及び並立的逐次実行単位に分類された逐次実行単位のうち停止された逐次実行単位の実行を再開する再開手段とをコンピュータを機能させるためのプログラムとして記録した記録媒体。

【請求項 3 5】 デバッグ対象のプログラムを構成する逐次実行単位のうち特定の逐次実行単位を対象として、該逐次実行単位の実行状態の制御を行うための制御指示を入力する特定制御指示入力手段と、

前記デバッグ対象のプログラムに含まれる各逐次実行単位を、被制御形態に基づいて排他的逐次実行単位、並立的逐次実行単位及び継続的逐次実行単位のいずれかに分類する分類手段と、

前記特定制御指示入力手段に逐次実行単位の実行を停止する指示が入力された場合には、前記分類手段により並立的逐次実行単位に分類された逐次実行単位のうち実行中の総ての逐次実行単位及び前記分類手段により排他的逐次実行単位に分類された逐次実行単位のうち前記特定制御指示入力手段によって第 1 の指定がされた逐次実行単位の実行を停止する停止手段と、

前記特定制御指示入力手段に逐次実行単位の実行を再開

10

20

30

40

50

する指示が入力された場合には、前記分類手段により並立的逐次実行単位に分類された逐次実行単位の総て、及び排他的逐次実行単位に分類された逐次実行単位のうち前記特定制御指示入力手段によって第 2 の指定がされた逐次実行単位の実行を再開する再開手段とをコンピュータを機能させるためのプログラムとして記録した記録媒体。

【請求項 3 6】 前記制御手段は、前記デバッグ対象プログラムに含まれる各逐次実行単位の状態指標に対応させて該逐次実行単位の実行状態の制御を行う手段を具備することを特徴とする請求項 3 3 記載の記録媒体。

【請求項 3 7】 内在的停止命令を含む命令群を逐次実行する特定の逐次実行単位を含むプログラムのデバッグを行うためのプログラムが記録された記録媒体において、前記特定の逐次実行単位に対する単位実行指示を入力するための特定制御指示入力手段と、前記特定制御指示入力手段に第 1 の単位実行指示が入力されたときに前記特定の逐次実行単位が次に実行すべき命令が前記内在的停止命令である場合には前記特定の逐次実行単位を実行停止状態とし、前記特定制御指示入力手段に前記実行停止状態とされた逐次実行単位に対しての第 2 の単位実行指示が入力されると該逐次実行単位が前記内在的停止命令の次の命令を実行可能な状態にする手段とをコンピュータを機能させるためのプログラムとして記録した記録媒体。

【請求項 3 8】 命令群を有するモジュールが定義されるデバッグ対象のプログラムに含まれる各モジュールを着目形態において分類する分類手段と、前記分類手段による着目形態に応じたデバッグ制御を行う手段とをコンピュータを機能させるためのプログラムとして記録した記録媒体。

【請求項 3 9】 前記分類手段は、前記モジュールを分類のための規則を用いて分類する手段と、外部からの指示を入力するための手段と、前記入力された指示に基づいて前記モジュールの分類を更新する手段とを具備することを特徴とする請求項 3 7 記載の記録媒体。

【請求項 4 0】 デバッグ対象のプログラム内のモジュールで定義される命令群に対する単位実行指示が入力される制御指示入力手段と、前記各モジュールを着目属性に応じて着目モジュール及び非着目モジュールに分類する分類手段と、前記制御指示入力手段に前記非着目モジュール内で定義された命令群への単位実行指示が入力された場合には該命令群全体を実行する手段とをコンピュータを機能させるためのプログラムとして記録した記録媒体。

【請求項 4 1】 デバッグ対象のプログラムに含まれる

並列して実行される逐次実行単位が実行する一、又は、複数の命令群を有するモジュールを分類するモジュール分類手段と、

前記各逐次実行単位を、前記モジュール分類手段による分類を考慮して定められる分類規則に基づいて前記デバッグ対象プログラムの実行時に動的に分類する逐次実行単位分類手段と、

前記逐次実行単位分類手段による分類情報に従ってデバッグ制御を行う手段とをさらに具備することを特徴とする請求項 2 5 または 2 6 記載の記録媒体。

【請求項 4 2】 並列して実行される逐次実行単位を含むデバッグ対象プログラムの動作を制御するための第 1 の制御指示群を記録する手段と、前記制御指示群に従って前記デバッグ対象プログラムの動作を制御することで得られる前記デバッグ対象プログラムの第 1 の動作過程を記録する動作過程記録手段と、前記動作過程記録手段に記録された第 1 の動作過程と同じ、又は、異なる動作をするように該各逐次実行単位を制御する第 2 の制御指示群を生成する手段とをコンピュータを機能させるためのプログラムとして記録した記録媒体。

【請求項 4 3】 前記生成された第 2 の制御指示群に従って前記デバッグ対象プログラムの動作を制御する手段と、前記実行によって得られる第 2 の動作過程と前記第 1 の動作過程とを比較する手段とをさらに具備することを特徴とする請求項 4 2 記載の記録媒体。

【請求項 4 4】 デバッグ対象のプログラムに含まれる並列して実行される逐次実行単位を属性に応じて分類する分類手段と、前記分類手段により特定の属性を持つと分類される逐次実行単位の動作状態を、該逐次実行単位ごとに設けられたウインドウに表示する手段と、

前記ウインドウごとに、該ウインドウに対応する逐次実行単位のみを対象とする第 1 の制御指示を入力するための手段と、

前記デバッグ対象プログラム全体を対象とする第 2 の制御指示を入力するための手段とをコンピュータを機能させるためのプログラムとして記録した記録媒体。

【請求項 4 5】 実行中の逐次実行単位の増減及び前記分類手段による逐次実行単位の分類の変化に対応させて前記ウインドウを前記デバッグ対象プログラムの実行時に動的に増減させる手段をさらに具備することを特徴とする請求項 4 4 記載の記録媒体。

【請求項 4 6】 デバッグ対象のプログラムに含まれる並列して実行される逐次実行単位を分類する手段と、前記逐次実行単位ごとに設けられたウインドウに該逐次実行単位の動作状態を前記分類に適応させて可視的に表示する手段と、

前記デバッグ対象プログラム全体を対象とする第 1 の制

御指示を入力するための手段と、
前記ウインドウごとに該ウインドウに対応する逐次実行
単位のみを対象とする第 2 の制御指示を入力するための
手段とをコンピュータを機能させるためのプログラムと
して記録した記録媒体。

【請求項 4 7】 デバッグ対象のプログラム内の並列実
行される逐次実行単位ごとに設けられたウインドウ画面
内に、

前記逐次実行単位に対する前記第 2 の制御指示を入力す
るための手段のための表示を該逐次実行単位の実行状態
に適応させて行う手段をコンピュータを機能させるため
のプログラムとして記録した記録媒体。

【請求項 4 8】 前記逐次実行単位ごとに設けられたウ
インドウ画面ごとに、

前記逐次実行単位が実行中のモジュールもしくは命令群
を表示する手段と、

該ウインドウに前記モジュールを構成する命令群を呼出
す論理軌跡を表示する手段とをさらに具備することを特
徴とする請求項 4 7 記載の記録媒体。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明はデバッグ装置、特に
例えばマルチスレッドプログラムを含む並列プログラ
ムのデバッグを効率良く行うことのできるデバッグ装置と
デバッグをコンピュータに実行させるためのプログラム
が記録された記憶媒体に関する。

【0002】

【従来の技術】従来より、プログラム中に含まれる誤り
(バグ) をプログラマーが取る作業 (デバッグ) を補助
する装置として、デバッグ装置なる装置が使われてき
た。デバッグ装置は、デバッグ作業からの指示に基づ
きプログラムの状態を制御することができる。

【0003】例えば、デバッグ装置を用いて、デバッグ
対象のプログラムを実行する際、そのソースプログラム
中のある行に、デバッグ作業がブレークポイントを指
定しておく、プログラムがその行を実行しようとする
直前に、デバッグ装置は、デバッグ対象のプログラムの
実行状態を停止状態にし、デバッグ作業の指示を待
つ。この時点で、デバッグ作業はデバッグ装置を介し
て、デバッグ対象のプログラムの状態 (プログラムが次
に実行しようとしている位置や各種の変数値等) を調査
できる。又、デバッグ作業は、この時点で、デバッグ
対象のプログラムの実行再開命令や、単位実行命令 (ス
テップ命令) 等のデバッグコマンドをデバッグ装置に与
えることができる。例えば、デバッグ作業がステップ
命令をデバッグ装置に指示すると、デバッグ装置は、デ
バッグ対象のプログラムを一旦実行状態に戻し、デバ
ッグ対象のプログラムが一つの命令を実行し終えた時点
で、再び、これを停止状態にし、デバッグ作業にデバ
ッグ対象のプログラムの状態を示す。ステップ命令を繰

り返してデバッグ装置に指示すれば、デバッグ作業は
デバッグ対象のプログラムがいかにしてソースプログラ
ム内を実行していくかの過程を追跡することができる。

【0004】このようにして、デバッグ装置を利用する
と、逐次プログラムのデバッグ効率は非常に高くなる。

【0005】一方、並列プログラムは、逐次プログラム
に比較して、タイミングに起因する誤りを含む可能性が
非常に大きい。並列プログラムの一つとしてマルチスレ
ッドプログラムを例にとる。例えば、二つのスレッドを有
するマルチスレッドプログラムにおいて、両スレッドに共
有変数が存在する場合、一方のスレッドが、その共有変
数に書き込みを行い、他方のスレッドが読み込みを行う
とする。この場合、しばしば、書き込み、読み込みの実
行順序が設計者の期待するものと異なってしまうことが
ある。

【0006】上記のような、マルチスレッドプログラ
ムのタイミングに起因するバグを取り除くために、従来の
デバッグ装置では、次の様な機能をデバッグ作業者に提
供していた。

【0007】まず、デバッグ装置がプログラムを実行し
ている任意の時点で、デバッグ作業は存在するスレッ
ドのリストを見ることができる。このリストの中から、
デバッグ作業は一つのスレッドを選び、それをカレン
トスレッドに指定する。それ以降、カレントスレッドが
変更される (デバッグ作業がデバッグ装置に変更を指
示する場合やデバッグ装置が自動的に変更する場合があ
る) 迄、そのカレントスレッドが、デバッグ作業による
ステップ命令等のデバッグコマンドの対象となる。即
ち、例えば、ステップ命令をデバッグ作業が指示する
と、カレントスレッドを含む総てのスレッドの実行が再
開される。その後、カレントスレッドが、一命令の実行
を終了すると、その直後に総てのスレッドの実行が停止
させられる。但し、カレントスレッド以外のスレッド
は、実行が再開されてから、停止する迄にどれだけすす
むかは、デバッグ作業には分からない。

【0008】図 20 は従来のデバッグ装置をグラフィカ
ルユーザインターフェイス (GUI) を用いて実現した
場合の該装置画面の一例を示す図である。同図におい
て、コントロールパネル 2001 はカレントスレッドの
実行をデバッグ作業が制御するためのパネルであり、
プログラム表示ウインドウ 2002 は、カレントスレッ
ドの実行位置をデバッグ作業に示すウインドウであ
る。コントロールパネル 2001 には、停止したカレン
トスレッドの実行を再開するための `Cont` ボタン 20
03、停止したカレントスレッドを一命令だけ実行させ
るための『`Step In`』ボタン 2004、停止したカ
レントスレッドを一命令だけ実行する際、関数呼び出し
に関しては、それを一命令として実行するための『`St
ep Over`』ボタン 2005、停止したカレントス
レッドの実行を再開し、現在の関数の実行を終えたとき

11

に停止させるための『Step Out』ボタン2006、プログラム表示ウィンドウ2002に表示するカレントスレッドのスタックフレームを一つ上げるための『Up』ボタン2007、プログラム表示ウィンドウ2002に表示するカレントスレッドのスタックフレームを一つ下げるための『Down』ボタン2008等が含まれる。

【0009】一方、プログラム表示ウィンドウ2002では、カレントスレッドの現在のスタックフレームに対応するソースプログラム2009が表示されている。位置ポインタ2010は、カレントスレッドが次に実行する位置を示すためのポインタであり、ブレークポイントポインタ2011はユーザが設定したブレークポイントを示すためのポインタである。

【0010】従来のデバッグ装置では、カレントスレッドの切替えは、その為のダイアログや、スレッド状態表示ウィンドウ等を用いてデバッグ作業者に指定させる場合が多い。図20に示された状態では、カレントスレッドは、関数f()内のステートメント「a=g();」を次に実行する状態で停止している。この状態でデバッグ作業者がStep Inボタン2004を押すと、カレントスレッド、及びその他のスレッドの実行が再開される。カレントスレッドが一行の実行を終えると、デバッグ装置はカレントスレッド、及び、その他のスレッドの実行を総て停止させる。又、カレントスレッドの実行中に、あるスレッドがブレークポイント2011の付けられた行に達すると、総てのスレッドの実行が停止し、デバッグ作業者が次の命令をコントロールパネル2001を用いてカレントスレッドに与えられる状態になる。ブレークポイントに達したスレッドがStep Inボタン2004ボタンが押下される前のカレントスレッドと異なる場合には、ブレークポイントに達したスレッドが新しくカレントスレッドになるという仕様を採り入れているデバッグ装置が多い。又、カレントスレッドの実行中に、そのカレントスレッドがサスペンドし、実行継続ができなくなった場合には、その他のスレッドがデバッグ装置によって自動的に選択されてカレントスレッドになるという仕様を採り入れているデバッグ装置が多い。

【0011】上記従来のデバッグ装置では、デバッグ作業者が特定のスレッドをカレントスレッドに指定し、それに対してステップ実行を繰り返している時、もし、その他のスレッドが偶然、バグの原因となるタイミングで実行された場合には、誤りの発見をすることは可能である。

【0012】しかしながら、デバッグ作業者は非カレントスレッドの実行のタイミングを任意に定めることができないため、誤りを起こす状況を自由に再現させることができない。複数のスレッドの実行タイミングをデバッグ作業者が制御し得るためには、ステップ実行の都度、

12

複数のスレッドに『suspend』命令や『resume』命令を与える必要があり、非常に煩雑な操作を強いられることになる。又、カレントスレッドがデバッグ作業者の意図しないときに切り替わることがあると、デバッグ作業者は混乱を来す恐れがある。

【0013】

【発明が解決しようとする課題】上述したように、従来のデバッグ装置においては、デバッグ作業者は非カレントスレッドの実行のタイミングを任意に定めることができないため、誤りを起こす状況を自由に再現させることができないという問題点があった。

【0014】また、複数のスレッドの実行タイミングをデバッグ作業者が制御し得るためには、ステップ実行の都度、複数のスレッドに『suspend』命令や『resume』命令を与える必要があり、非常に煩雑な操作を強いられることになるという問題点があった。

【0015】また、カレントスレッドがデバッグ作業者の意図しないときに切り替わることがある場合、デバッグ作業者は混乱を来す恐れがあるという点も問題となっていた。

【0016】なお、上記の問題点は、マルチスレッドプログラムの場合に対するものを例示的に挙げたものであり、並列プログラム、又は、並列プログラム一般に同様の問題点が指摘される。

【0017】本発明は、このような従来技術上の問題点を解決するためになされたものであり、効率的なデバッグを可能とするデバッグ装置及び記録媒体を提供することを各請求項に記載された発明の共通した目的とする。

【0018】さらに、各請求項に記載された発明は、複数のスレッドの実行を、デバッグ作業者の希望する順序で実行せしめ、タイミングに起因するバグの再現を容易に行うことを可能とするデバッグ装置及び記録媒体を提供することを目的とする。

【0019】各請求項に記載された発明ごとには、以下の課題が目的として挙げられる。

【0020】請求項1に係る本願発明は、逐次実行単位間の実行のタイミングを任意に制御することにより、より効率的なデバッグを可能とするデバッグ装置の提供を目的とする。

【0021】請求項2に係る本願発明は、逐次実行単位間の再開、停止のタイミングを任意に制御することにより、より効率的なデバッグを可能とするデバッグ装置を提供することを目的とする。

【0022】請求項3に係る本願発明は、請求項1及び2記載の発明の目的とするところに加え、逐次実行単位の実行動作の排他性・従属性の度合いに応じて異なる制御をすることにより、より効率的なデバッグを可能とするデバッグ装置を提供することを目的とする。

【0023】請求項4に係る本願発明は、請求項3記載の発明の目的とするところに加え、逐次実行単位に応じ

10

20

30

40

50

てデバッグ中に異なる制御をすることにより、より効率的なデバッグを可能とするデバッグ装置を提供することを目的とする。

【0024】請求項5に係る本願発明は、請求項3及び4記載の発明の目的とするところに加え、逐次実行単位の実行動作の排他性・従属性の度合いに応じてデバッグ制御を行うことにより、より効率的なデバッグを可能とするデバッグ装置を提供することを目的とする。

【0025】請求項6に係る本願発明は、請求項3及び4記載の発明の目的とするところに加え、逐次実行単位の10 実行動作の着目性の度合いに応じてデバッグ制御を行うことにより、より効率的なデバッグを可能とするデバッグ装置を提供することを目的とする。

【0026】請求項7に係る本願発明は、請求項3及び4記載の発明の目的とするところに加え、逐次実行単位の10 実行動作の表示性の度合いに応じてデバッグ制御を行うことにより、より効率的なデバッグを可能とするデバッグ装置を提供することを目的とする。

【0027】請求項8に係る本願発明は、デバッグ制御において特定の逐次実行単位のみを対象とする制御指示と、プログラム全体を対象とする制御指示を分けることにより、特定の逐次実行単位にのみ注目してデバッグ制御することで、より効率的なデバッグを可能にするデバッグ装置を提供することを目的とする。

【0028】請求項9乃至11に係る本願発明は、請求項1及び2に係る本願発明の目的とするところ及び請求項8に係る本願発明の目的とするところを同時に達成すること、即ち、全体のプログラム／特定の逐次実行単位を対象とする一般／特定制御指示により他の排他的逐次実行単位の実行を停止させる間に微妙なタイミングの制御をすることにより、より効率的なデバッグを可能にするデバッグ装置を提供することを目的とする。

【0029】請求項12に係る本願発明は、請求項9に係る本願発明の目的とするところに加えて、逐次実行単位の動作状態をデバッグ制御に的確に反映させることで、実際には起こり得ないタイミングの実行を回避し、より効率的なデバッグを可能にするデバッグ装置を提供することを目的とする。

【0030】請求項13に係る本願発明は、請求項9に係る本願発明の目的とするところに加えて、逐次実行単位の10 実行の微妙なタイミングを制御することにより、より効率的なデバッグを可能にするデバッグ装置を提供することを目的とする。

【0031】請求項14に係る本願発明は、そのプログラムに含まれるモジュールをその属性に応じてデバッグ制御において差別的に取り扱うことにより、より効率的なデバッグを可能にするデバッグ装置を提供することを目的とする。

【0032】請求項15に係る本願発明は、請求項14記載の発明の目的とするところに加えて、そのモジュー10

ルの属性をデバッグ作業者が変更することにより、より効率的なデバッグを可能にするデバッグ装置を提供することを目的とする。

【0033】請求項16に係る本願発明は、着目の対象でないモジュールに対する容易な実行制御を可能にすることにより、より効率的なデバッグを可能とするデバッグ装置を提供することを目的とする。

【0034】請求項17に係る本願発明は、請求項1及び2記載の発明の目的とするところと請求項16記載の発明の目的とするところを同時に達成することを可能にするデバッグ装置を提供することを目的とする。

【0035】請求項18および19に係る本願発明は、逐次実行単位の実行のタイミングを制御した上でデバッグ制御の自動実行をすることにより、さらに効率的なデバッグ及び自動テストを可能にするデバッグ装置を提供することを目的とする。

【0036】請求項20乃至22に係る本願発明は、並列プログラムをデバッグ装置を用いて実行する際に、各逐次実行単位の動作状態を並列的に可視化することを可能にするデバッグ装置を提供することを目的とする。

【0037】請求項23に係る本願発明は、逐次実行単位の動作状態を、デバッグ制御の入力手段の表示形態に反映させることにより、さらに効率的なデバッグを可能にするデバッグ装置を提供することを目的とする。

【0038】請求項24に係る本願発明は、請求項23記載の発明の目的とするところに加えて、モジュールの呼出軌跡を確認しながらデバッグ作業を進めることを可能にするデバッグ装置を提供することを目的とする。

【0039】さらに、請求項25乃至48に係る本発明は、以上の課題を同様に解決し得る機能をコンピュータにおいて実現するプログラムが記録された記録媒体の提供を目的とする。

【0040】

【課題を解決するための手段】かかる目的を達成するために、請求項1記載の発明のデバッグ装置は、並列して実行される逐次実行単位を含むプログラムについてのデバッグの制御を行う制御手段と、前記逐次実行単位の内、特定の逐次実行単位のみについて前記制御手段を動作させる対象特定制御手段とを具備する。

【0041】請求項2記載の発明は、請求項1記載の発明のデバッグ装置において、前記対象特定制御手段が、デバッグ制御の対象となる特定の逐次実行単位のみについて実行の再開または停止を行わせる手段をさらに具備する。

【0042】請求項3記載の本発明は、請求項1または2記載のデバッグ装置において、前記対象特定制御手段は、前記各逐次実行単位の分類を指定するための手段と、前記指定された分類を表す分類情報を入力するための手段と、前記入力された分類情報に従ってデバッグ制御を行う手段とを具備する。

【0043】請求項4記載の本発明は、請求項3記載のデバッグ装置において、前記対象特定制御手段は、前記各逐次実行単位の分類のための規則を記録する手段と、前記記録された規則を用いて前記各逐次実行単位の初期の分類を定める手段と、前記分類を変更するための手段とをさらに具備する。

【0044】請求項5記載の本発明は、請求項3または4記載のデバッグ装置において、前記分類情報は、デバッグ装置が前記逐次実行単位を制御する形態に関する情報であることを特徴とする。

【0045】請求項6記載の本発明は、請求項3または4記載のデバッグ装置において、前記分類情報は、前記逐次実行単位についてデバッグ作業者が着目する程度に関する情報であることを特徴とする。

【0046】請求項7記載の本発明は、請求項3または4記載のデバッグ装置において、前記分類情報は、デバッグ装置が前記逐次実行単位の状態を表示させる形態に関する情報であることを特徴とする。

【0047】請求項8記載の発明は、並列して実行される逐次実行単位を含むプログラムをデバッグするデバッグ装置において、前記デバッグ装置は、前記プログラムの状態を制御する指示を入力するための手段を具備し、前記指示は、前記プログラムの状態のうち、特定の逐次実行単位に関する実行状態のみを対象とする特定制御指示と、前記プログラムの状態全体を対象とする一般制御指示とを含むことを特徴とする。

【0048】請求項9記載の発明のデバッグ装置は、デバッグ対象のプログラムの状態全体を対象とする制御を行うための制御指示を入力する一般制御指示入力手段と、前記デバッグ対象のプログラムに含まれる特定の逐次実行単位を対象として、該逐次実行単位の実行状態の制御を行うための制御指示を入力する特定制御指示入力手段と、前記デバッグ対象のプログラムに含まれる各逐次実行単位を分類する分類手段と、前記一般制御指示入力手段もしくは特定制御指示入力手段にプログラムの状態を制御するための指示が入力された場合には、前記分類手段による分類に基づく制御情報に従ってプログラムの状態の制御を行う制御手段とを具備する。

【0049】請求項10記載の発明のデバッグ装置は、デバッグ対象のプログラムの状態全体を対象とする制御を行うための制御指示を入力する一般制御指示入力手段と、前記デバッグ対象のプログラムに含まれる各逐次実行単位を、被制御形態に基づいて排他的逐次実行単位、並立的逐次実行単位及び継続的逐次実行単位のいずれかに分類する分類手段と、前記一般制御指示入力手段に逐次実行単位の実行を停止する指示が入力された場合には、前記分類手段により排他的逐次実行単位及び並立的逐次実行単位に分類された逐次実行単位のうち実行中の逐次実行単位の実行を停止する停止手段と、前記一般制御指示入力手段に逐次実行単位の実行を再開する指示が

入力された場合には、前記分類手段により排他的逐次実行単位及び並立的逐次実行単位に分類された逐次実行単位のうち停止された逐次実行単位の実行を再開する再開手段とを具備する。

【0050】請求項11記載の発明のデバッグ装置は、デバッグ対象のプログラムを構成する逐次実行単位のうち特定の逐次実行単位を対象として、該逐次実行単位の実行状態の制御を行うための制御指示を入力する特定制御指示入力手段と、前記デバッグ対象のプログラムに含まれる各逐次実行単位を、被制御形態に基づいて排他的逐次実行単位、並立的逐次実行単位及び継続的逐次実行単位のいずれかに分類する分類手段と、前記特定制御指示入力手段に逐次実行単位の実行を停止する指示が入力された場合には、前記分類手段により並立的逐次実行単位に分類された逐次実行単位のうち実行中の総ての逐次実行単位及び前記分類手段により排他的逐次実行単位に分類された逐次実行単位のうち前記特定制御指示入力手段によって第1の指定がされた逐次実行単位の実行を停止する停止手段と、前記特定制御指示入力手段に逐次実行単位の実行を再開する指示が入力された場合には、前記分類手段により並立的逐次実行単位に分類された逐次実行単位の総て、及び排他的逐次実行単位に分類された逐次実行単位のうち前記特定制御指示入力手段によって第2の指定がされた逐次実行単位の実行を再開する再開手段とを具備する。

【0051】請求項12記載の発明は、請求項9記載のデバッグ装置において、前記制御手段は、前記デバッグ対象のプログラムに含まれる各逐次実行単位の状態指標に対応させて該逐次実行単位の実行状態の制御を行う手段を具備するものである。

【0052】請求項13記載の発明は、内在的停止命令を含む命令群を逐次実行する特定の逐次実行単位を含むプログラムのデバッグ装置において、前記特定の逐次実行単位に対する単位実行指示を入力するための特定制御指示入力手段と、前記特定制御指示入力手段に第1の単位実行指示が入力されたときに前記特定の逐次実行単位が次に実行すべき命令が前記内在的停止命令である場合には前記特定の逐次実行単位を実行停止状態とし、前記特定制御指示入力手段に前記実行停止状態とされた逐次実行単位に対しての第2の単位実行指示が入力されると該逐次実行単位が前記内在的停止命令の次の命令を実行可能な状態にする手段とを具備する。

【0053】請求項14記載の発明は、命令群を有するモジュールが定義されるデバッグ対象のプログラムに含まれる各モジュールを着目形態において分類する分類手段と、前記分類手段による着目形態に応じたデバッグ制御を行う手段とを具備する。

【0054】請求項15記載の発明は、請求項14記載のデバッグ装置において、前記分類手段は、前記モジュールを分類のための規則を用いて分類する手段と、外部

からの指示を入力するための手段と、前記入力された指示に基づいて前記モジュールの分類を更新する手段とを具備する。

【0055】請求項16記載の発明のデバッグ装置は、デバッグ対象のプログラム内のモジュールで定義される命令群に対する単位実行指示が入力される制御指示入力手段と、前記各モジュールを着目属性に応じて着目モジュール及び非着目モジュールに分類する分類手段と、前記制御指示入力手段に前記非着目モジュール内で定義された命令群への単位実行指示が入力された場合には該命令群全体を実行する手段とを具備する。

【0056】請求項17記載の発明は、請求項1または2記載のデバッグ装置において、デバッグ対象のプログラムに含まれる並列して実行される逐次実行単位が実行する一、又は、複数の命令群を有するモジュールを分類するモジュール分類手段と、前記各逐次実行単位を、前記モジュール分類手段による分類を考慮して定められる分類規則に基づいてプログラムの実行時に動的に分類する逐次実行単位分類手段と、前記逐次実行単位分類手段による分類情報に従ってデバッグ制御を行う手段とをさらに具備するものである。

【0057】請求項18記載の発明のデバッグ装置は、並列して実行される逐次実行単位を含むプログラムの動作を制御するための第1の制御指示群を記録する手段と、前記制御指示群に従って前記プログラムの動作を制御することで得られる前記プログラムの第1の動作過程を記録する動作過程記録手段と、前記動作過程記録手段に記録された第1の動作過程と同じ、又は、異なる動作をするように該各逐次実行単位を制御する第2の制御指示群を生成する手段とを具備する。

【0058】請求項19記載の発明は、請求項18記載のデバッグ装置において、前記生成された第2の制御指示群に従って前記プログラムの動作を制御する手段と、前記実行によって得られる第2の動作過程と前記第1の動作過程とを比較する手段とをさらに具備してなるものである。

【0059】請求項20記載の発明のデバッグ装置は、デバッグ対象のプログラムに含まれる並列して実行される逐次実行単位を属性に応じて分類する分類手段と、前記分類手段により特定の属性を持つと分類される逐次実行単位の動作状態を、該逐次実行単位ごとに設けられたウインドウに表示する手段と、前記ウインドウごとに、該ウインドウに対応する逐次実行単位のみを対象とする第1の制御指示を入力するための手段と、前記プログラム全体を対象とする第2の制御指示を入力するための手段とを具備する。

【0060】請求項21記載の発明は、請求項20記載のデバッグ装置において、実行中の逐次実行単位の増減及び前記分類手段による逐次実行単位の分類の変化に対応させて前記ウインドウをプログラムの実行時に動的に

増減させる手段をさらに具備してなる。

【0061】請求項22記載の発明のデバッグ装置は、デバッグ対象のプログラムに含まれる並列して実行される逐次実行単位を分類する手段と、前記逐次実行単位ごとに設けられたウインドウに該逐次実行単位の動作状態を前記分類に適応させて可視的に表示する手段と、前記プログラム全体を対象とする第1の制御指示を入力するための手段と、前記ウインドウごとに該ウインドウに対応する逐次実行単位のみを対象とする第2の制御指示を入力するための手段とを具備してなる。

【0062】請求項23記載の発明のデバッグ装置は、デバッグ対象のプログラム内の並列実行される逐次実行単位ごとに設けられたウインドウ画面内に、前記逐次実行単位に対する前記第2の制御指示を入力するための手段のための表示を該逐次実行単位の実行状態に適応させて行う手段を具備する。

【0063】請求項24記載の発明は、請求項23記載のデバッグ装置において、前記逐次実行単位ごとに設けられたウインドウ画面ごとに、前記逐次実行単位が実行中のモジュールもしくは命令群を表示する手段と、該ウインドウに前記モジュールを構成する命令群を呼出す論理軌跡を表示する手段とをさらに具備する。

【0064】請求項25記載の発明の記録媒体は、並列して実行される逐次実行単位を含むデバッグ対象プログラムについてのデバッグの制御を行う制御手段と、前記逐次実行単位のうち特定の逐次実行単位のみについて前記制御手段を動作させる対象特定制御手段とをコンピュータを機能させるためのプログラムとして記録してなるものである。

【0065】請求項26記載の発明の記録媒体は、請求項25記載の記録媒体において、前記対象特定制御手段が、デバッグ制御の対象となる特定の逐次実行単位のみについて実行の再開または停止を行わせる手段をさらに具備することを特徴とするものである。

【0066】請求項27記載の発明の記録媒体は、請求項25または26記載の記録媒体において、前記対象特定制御手段は、前記各逐次実行単位の分類を指定するための手段と、前記指定された分類を表す分類情報を入力するための手段と、前記入力された分類情報に従ってデバッグ制御を行う手段とを具備することを特徴とするものである。

【0067】請求項28記載の発明の記録媒体は、請求項27記載の記録媒体において、前記対象特定制御手段は、前記各逐次実行単位の分類のための規則を記録する手段と、前記記録された規則を用いて前記各逐次実行単位の初期の分類を定める手段と、前記分類を変更するための手段とをさらに具備することを特徴とするものである。

【0068】請求項29記載の発明の記録媒体は、請求項27または28記載の記録媒体において、前記分類情

報は、前記逐次実行単位を制御する形態に関する情報であることを特徴とするものである。

【0069】請求項30記載の発明の記録媒体は、請求項27または28記載の記録媒体において、前記分類情報は、前記逐次実行単位についてデバッグ作業者が着目する程度に関する情報であることを特徴とするものである。

【0070】請求項31記載の発明の記録媒体は、請求項27または28記載の記録媒体において、前記分類情報は、前記逐次実行単位の状態を表示させる形態に関する情報であることを特徴とするものである。

【0071】請求項32記載の発明の記録媒体は、並列して実行される逐次実行単位を含むデバッグ対象プログラムの状態を制御する指示を入力するための手段がコンピュータを機能させるためのプログラムとして記録され、前記指示は、前記デバッグ対象プログラムの状態のうち、特定の逐次実行単位に関する実行状態のみを対象とする特定制御指示と、前記デバッグ対象プログラムの状態全体を対象とする一般制御指示とを含むことを特徴とするものである。

【0072】請求項33記載の発明の記録媒体は、デバッグ対象プログラムの状態全体を対象とする制御を行うための制御指示を入力する一般制御指示入力手段と、前記デバッグ対象プログラムに含まれる特定の逐次実行単位を対象として、該逐次実行単位の実行状態の制御を行うための制御指示を入力する特定制御指示入力手段と、前記デバッグ対象プログラムに含まれる各逐次実行単位を分類する分類手段と、前記一般制御指示入力手段もしくは特定制御指示入力手段に前記デバッグ対象プログラムの状態を制御するための指示が入力された場合には、前記分類手段による分類に基づく制御情報に従って前記デバッグ対象プログラムの状態の制御を行う制御手段とをコンピュータを機能させるためのプログラムとして記録したものである。

【0073】請求項34記載の発明の記録媒体は、デバッグ対象のプログラムの状態全体を対象とする制御を行うための制御指示を入力する一般制御指示入力手段と、前記デバッグ対象プログラムに含まれる各逐次実行単位を、被制御形態に基づいて排他的逐次実行単位、並立的逐次実行単位及び継続的逐次実行単位のいずれかに分類する分類手段と、前記一般制御指示入力手段に逐次実行単位の実行を停止する指示が入力された場合には、前記分類手段により排他的逐次実行単位及び並立的逐次実行単位に分類された逐次実行単位のうち実行中の逐次実行単位の実行を停止する停止手段と、前記一般制御指示入力手段に逐次実行単位の実行を再開する指示が入力された場合には、前記分類手段により排他的逐次実行単位及び並立的逐次実行単位に分類された逐次実行単位のうち停止された逐次実行単位の実行を再開する再開手段とをコンピュータを機能させるためのプログラムとして記録

したものである。

【0074】請求項35記載の発明の記録媒体は、デバッグ対象のプログラムを構成する逐次実行単位のうち特定の逐次実行単位を対象として、該逐次実行単位の実行状態の制御を行うための制御指示を入力する特定制御指示入力手段と、前記デバッグ対象のプログラムに含まれる各逐次実行単位を、被制御形態に基づいて排他的逐次実行単位、並立的逐次実行単位及び継続的逐次実行単位のいずれかに分類する分類手段と、前記特定制御指示入力手段に逐次実行単位の実行を停止する指示が入力された場合には、前記分類手段により並立的逐次実行単位に分類された逐次実行単位のうち実行中の総ての逐次実行単位及び前記分類手段により排他的逐次実行単位に分類された逐次実行単位のうち前記特定制御指示入力手段によって第1の指定がされた逐次実行単位の実行を停止する停止手段と、前記特定制御指示入力手段に逐次実行単位の実行を再開する指示が入力された場合には、前記分類手段により並立的逐次実行単位に分類された逐次実行単位の総て、及び排他的逐次実行単位に分類された逐次実行単位のうち前記特定制御指示入力手段によって第2の指定がされた逐次実行単位の実行を再開する再開手段とをコンピュータを機能させるためのプログラムとして記録したものである。

【0075】請求項36記載の発明の記録媒体は、請求項33記載の記録媒体において、前記制御手段は、前記デバッグ対象プログラムに含まれる各逐次実行単位の状態指標に対応させて該逐次実行単位の実行状態の制御を行う手段を具備することを特徴とするものである。

【0076】請求項37記載の発明の記録媒体は、内在的停止命令を含む命令群を逐次実行する特定の逐次実行単位を含むプログラムのデバッグを行うためのプログラムが記録された記録媒体において、前記特定の逐次実行単位に対する単位実行指示を入力するための特定制御指示入力手段と、前記特定制御指示入力手段に第1の単位実行指示が入力されたときに前記特定の逐次実行単位が次に実行すべき命令が前記内在的停止命令である場合には前記特定の逐次実行単位を実行停止状態とし、前記特定制御指示入力手段に前記実行停止状態とされた逐次実行単位に対しての第2の単位実行指示が入力されると該逐次実行単位が前記内在的停止命令の次の命令を実行可能な状態にする手段とをコンピュータを機能させるためのプログラムとして記録したものである。

【0077】請求項38記載の発明の記録媒体は、命令群を有するモジュールが定義されるデバッグ対象のプログラムに含まれる各モジュールを着目形態において分類する分類手段と、前記分類手段による着目形態に応じたデバッグ制御を行う手段とをコンピュータを機能させるためのプログラムとして記録したものである。

【0078】請求項39記載の発明の記録媒体は、請求項37記載の記録媒体において、前記分類手段は、前記

モジュールを分類のための規則を用いて分類する手段と、外部からの指示を入力するための手段と、前記入力された指示に基づいて前記モジュールの分類を更新する手段とを具備することを特徴とするものである。

【0079】請求項40記載の発明の記録媒体は、デバッグ対象のプログラム内のモジュールで定義される命令群に対する単位実行指示が入力される制御指示入力手段と、前記各モジュールを着目属性に応じて着目モジュール及び非着目モジュールに分類する分類手段と、前記制御指示入力手段に前記非着目モジュール内で定義された命令群への単位実行指示が入力された場合には該命令群全体を実行する手段とをコンピュータを機能させるためのプログラムとして記録したものである。

【0080】請求項41記載の発明の記録媒体は、請求項25または26記載の記録媒体において、デバッグ対象のプログラムに含まれる並列して実行される逐次実行単位が実行する一、又は、複数の命令群を有するモジュールを分類するモジュール分類手段と、前記各逐次実行単位を、前記モジュール分類手段による分類を考慮して定められる分類規則に基づいて前記デバッグ対象プログラムの実行時に動的に分類する逐次実行単位分類手段と、前記逐次実行単位分類手段による分類情報に従ってデバッグ制御を行う手段とをさらに具備することを特徴とするものである。

【0081】請求項42記載の発明の記録媒体は、並列して実行される逐次実行単位を含むデバッグ対象プログラムの動作を制御するための第1の制御指示群を記録する手段と、前記制御指示群に従って前記デバッグ対象プログラムの動作を制御することで得られる前記デバッグ対象プログラムの第1の動作過程を記録する動作過程記録手段と、前記動作過程記録手段に記録された第1の動作過程と同じ、又は、異なる動作をするように該各逐次実行単位を制御する第2の制御指示群を生成する手段とをコンピュータを機能させるためのプログラムとして記録したものである。

【0082】請求項43記載の発明の記録媒体は、請求項42記載の記録媒体において、前記生成された第2の制御指示群に従って前記デバッグ対象プログラムの動作を制御する手段と、前記実行によって得られる第2の動作過程と前記第1の動作過程とを比較する手段とをさらに具備することを特徴とするものである。

【0083】請求項44記載の発明の記録媒体は、デバッグ対象のプログラムに含まれる並列して実行される逐次実行単位を属性に応じて分類する分類手段と、前記分類手段により特定の属性を持つと分類される逐次実行単位の動作状態を、該逐次実行単位ごとに設けられたウィンドウに表示する手段と、前記ウィンドウごとに、該ウィンドウに対応する逐次実行単位のみを対象とする第1の制御指示を入力するための手段と、前記デバッグ対象プログラム全体を対象とする第2の制御指示を入力する

ための手段とをコンピュータを機能させるためのプログラムとして記録したものである。

【0084】請求項45記載の発明の記録媒体は、請求項44記載の記録媒体において、実行中の逐次実行単位の増減及び前記分類手段による逐次実行単位の分類の変化に対応させて前記ウィンドウを前記デバッグ対象プログラムの実行時に動的に増減させる手段をさらに具備することを特徴とするものである。

【0085】請求項46記載の発明の記録媒体は、デバッグ対象のプログラムに含まれる並列して実行される逐次実行単位を分類する手段と、前記逐次実行単位ごとに設けられたウィンドウに該逐次実行単位の動作状態を前記分類に適応させて可視的に表示する手段と、前記デバッグ対象プログラム全体を対象とする第1の制御指示を入力するための手段と、前記ウィンドウごとに該ウィンドウに対応する逐次実行単位のみを対象とする第2の制御指示を入力するための手段とをコンピュータを機能させるためのプログラムとして記録したものである。

【0086】請求項47記載の発明の記録媒体は、デバッグ対象のプログラム内の並列実行される逐次実行単位ごとに設けられたウィンドウ画面内に、前記逐次実行単位に対する前記第2の制御指示を入力するための手段のための表示を該逐次実行単位の実行状態に適応させて行う手段をコンピュータを機能させるためのプログラムとして記録したものである。

【0087】請求項48記載の発明の記録媒体は、請求項47記載の記録媒体において、前記逐次実行単位ごとに設けられたウィンドウ画面ごとに、前記逐次実行単位が実行中のモジュールもしくは命令群を表示する手段と、該ウィンドウに前記モジュールを構成する命令群を呼出す論理軌跡を表示する手段とをさらに具備することを特徴とするものである。

【0088】以上各請求項に記載される「デバッグ」とは、作成されたプログラム、又は、既存のプログラム中に含まれる論理的、又は、タイミング的な誤りを取り除く作業を示す概念である。「デバッグ装置」とは、デバッグ作業者を補助し、デバッグ作業者の作業量を軽減することを目的とする装置である。デバッグ装置は、「ソースコードレベルのデバッグ装置」、「機械語レベルのデバッグ装置」の二種類に大別される。「ソースコードレベルのデバッグ装置」とは、デバッグ対象のプログラムの実行状態を、そのプログラムのソースコード上の実行位置に対応させてデバッグ作業者に示す装置である。又、「機械語レベルのデバッグ装置」とは、プログラムの実行状態を、そのプログラムの機械語上の実行位置に対応させてデバッグ作業者に示す装置である。本発明における「デバッグ装置」は、「ソースコードレベルのデバッグ装置」、及び、「機械語レベルのデバッグ装置」の両者を指す。そして、本願発明で「ソースコード」と記述した場合、これは、「機械語レベルのデバッグ装

置」においては、機械語のソースコード、すなわち、デバッグ対象のプログラムのアセンブルコードを意味する。

【0089】「逐次実行単位」とは、並列プログラムにおける逐次実行単位全般を示す概念であり、例えば、マルチスレッドプログラムにおけるスレッド、マルチプロセスプログラムにおけるプロセス、マルチタスクプログラムにおけるタスク等を含む。本発明において「並列プログラム」というときは、特記ない限り、一または複数の逐次実行単位よりなるものとする。

【0090】「デバッグ対象のプログラムの実行状態」とは、デバッグ対象のプログラムを実行するCPUの状態、各種変数値を保存しているメモリの状態等を指す概念である。従って、「デバッグ対象の並列プログラムの実行状態」とは、デバッグ対象の並列プログラムを構成する各逐次実行単位に割当てられたCPUの状態、各逐次実行単位が固有に持つ変数を保存しているメモリの状態、複数の逐次実行単位の間で共有されている変数を保存しているメモリの状態等を意味する。

【0091】「デバッグ制御」とは、デバッグ装置によるデバッグ対象のプログラムの実行状態に対する制御を意味する。デバッグ制御は、デバッグ装置に対して、デバッグ作業者が指示を与えることによって行われる場合、及び、デバッグ装置が、デバッグ対象のプログラムの実行状態に応じて、自動的に行う場合がある。

【0092】「特定の逐次実行単位のみについて実行を再開または停止させる」とは、逐次実行単位のあるものが停止し、あるものが実行されている状態から、一部の逐次実行単位のみを再開または停止させるというデバッグ制御を示す概念であり、例えば、1つのスレッドの実行を停止または再開させるときに、他の停止し、または実行されているスレッドの実行状態を変更しないように論理設計されたプログラムや回路等によって実現される。

【0093】「分類」とは、逐次実行単位の被制御形態に応じて分類することをいう。ここで、「被制御形態」とは、例えばマルチスレッドプログラムのデバッグを行うに当たってデバッグ対象について用いられる属性を表す概念であり、実行形態、着目形態、表示形態の3つの形態も含まれる。また「実行形態」とは、デバッグ対象のプログラムに含まれる前記逐次実行単位の各逐次実行単位について、デバッグ過程でどう実行させ、及び停止させるかを表す性質を示す概念であり、例えばデバッグ対象のマルチスレッドプログラムに含まれる各スレッドを、本発明の対象たるデバッグ装置がどのように実行及び停止するのかを表すように、排他実行スレッド、同時実行スレッド、及び継続実行スレッドのいずれかに分類するときの属性が含まれる。

【0094】「着目形態」とは、デバッグ対象のプログラムに含まれる各逐次実行単位に対してデバッグ過程で

デバッグ作業者が着目する度合いを示す概念であり、例えばデバッグ対象のマルチスレッドプログラムに含まれる各スレッドを、着目すべき着目スレッド及び着目しない非着目スレッドのいずれかに分類するときの属性が含まれる。

【0095】「排他的逐次実行単位」とは、デバッグ作業者がデバッグ中に、そのタイミング等の振舞いを制御しようと思う逐次実行単位、例えばスレッドについて、かかるスレッドのみを排他的に実行させ他の排他的逐次実行単位の実行状態を変えないようにさせる場合の該スレッドを表す概念である。具体的にはデバッグ作業者がスレッド間の実行のタイミングを再現させたいと考えているスレッドである。

【0096】「並立的逐次実行単位」とは、デバッグ作業者がデバッグ中に、そのタイミング等の振舞いを制御しようと思わない逐次実行単位、例えばスレッドについて、かかるスレッドは他と並立的に実行させる場合の、該スレッドを表す概念であり、具体的には例えば、マルチスレッドプログラムに対する従来のデバッグにおける非カレントスレッドに相当するものである。

【0097】「継続的逐次実行単位」とは、デバッグ作業者がデバッグ中に、その着目の意思にかかわらず継続的な実行を要すると考える逐次実行単位、例えばスレッドについて、かかるスレッドは他と関係なく継続的に実行させるとする場合の、該スレッドを表す概念であり、例えば、GUIのイベント処理を行うスレッドのようなシステムスレッドが該当する。

【0098】「逐次実行単位の分類のための規則」とは、逐次実行単位を属性によって分類するための指標となるものを示す概念であり、例えば、デバッグ対象のマルチスレッドプログラムに含まれる各スレッドを、排他的実行スレッド、同時実行スレッド及び継続実行スレッドのいずれかに分類し、同じく各スレッドを着目スレッド及び非着目スレッドのいずれかに分類するために対応表を論理設計したプログラムや回路等によって実現される。

【0099】「一般制御指示」とは、デバッグ対象の並列プログラムを構成する特定逐次実行単位を対象せず、プログラム全体を対象とする制御指示を示す概念であり、例えば、デバッグ対象となるプログラムの指定、そのプログラムの実行の開始、終了や、デバッグ対象のプログラムに含まれる総てのスレッドに対して、実行再開、停止、ブレークポイント設定等を指示する一般デバッグコマンドが含まれる。

【0100】「特定制御指示」とは、デバッグ対象の並列プログラムを構成する特定逐次実行単位のみを対象とする制御指示を示す概念であり、例えば、デバッグ対象のマルチスレッドプログラムに含まれる特定のスレッドに対して、実行再開、停止、1ステップのみ実行、ブレークポイント設定等を指示するスレッド別デバッグコ

10

20

30

40

50

マンドが含まれる。スレッド別デバッグコマンドには、上に挙げたステップ命令の他に、実行中のスレッドを停止させる『Stop』等が含まれる。

【0101】「逐次実行単位の状態指標」とは、デバッグ対象プログラムの実行制御に反映されるべき逐次実行単位の本来の挙動のエミュレーションを示す概念であり、「逐次実行単位の優先度」をも包含する概念である。ここで、「逐次実行単位の優先度」とは、デバッグ対象プログラムの実行制御に反映されるべき逐次実行単位の実行の優劣を示す概念であり、例えば、デバッグ対象のマルチスレッドプログラムに包含されるスレッドt1及びt2に対して、スレッドt1の優先度がスレッドt2の優先度より高く設定されていれば、OSはスレッドt1に対してより優先的にCPUを割り当てる。従って、例えば、デバッグ対象のマルチスレッドプログラムに包含されるスレッドt1の優先度がスレッドt2の優先度よりも高く設定されているような場合も、本来の挙動は、スレッドt1の実行中にはスレッドt2の実行が抑制されるといったものとなる。

【0102】「内在的停止命令」とは、逐次実行単位の実行を、デバッグ装置による制御に起因してでなくデバッグ対象のプログラムが実行に伴って内在的に停止状態となる処理命令を示す概念である。例えば、デバッグ対象のプログラムに含まれる『sleep』命令をデバッグ対象のあるスレッドが実行すると、そのスレッドの実行が一定時間休眠状態になるように論理設計されたプログラムや回路等によって実現される。『sleep』以外にも、『suspend』、『wait』等の各種命令が存在する。

【0103】「単位実行指示」とは、この指示がデバッグ装置に与えられると、デバッグ装置は逐次実行単位に対して、実行を一定の単位だけ行ったのちに停止状態にするように制御するという指示である。例えば、『Step In』命令を実行すると、ソースコード上の1行のみ実行した後に停止状態に移行するように論理設計されたプログラムや回路等によって実現される。

【0104】「実行停止状態」とは、CPUがデバッグ対象のプログラムを実行するのを、デバッグ装置が一時的に中断させている状態である。例えば、デバッグ装置の制御の元で、デバッグ対象のプログラムが実行されているとき、このプログラムがブレークポイントの置かれた位置を実行しようとする、デバッグ装置は、このプログラムを実行停止状態に移行させる。

【0105】「命令群」とは、プログラムを構成する処理の集まりを示す概念であり、関数、メソッド、手続きなどの概念を含むものである。命令群は複数の処理命令によって構成される。ここで「処理命令」とは、例えば変数への代入命令、入出力命令等のプログラムを構成する基本単位を指す。

【0106】「モジュール」とは、一または複数の命令

群から構成される単位を示す概念であり、具体的には、オブジェクト指向言語におけるクラス、プログラムを構成するファイル、複数のファイルの論理的集まりであるパッケージなどを指す。

【0107】「モジュール分類手段による分類を考慮して定められる分類規則」とは、モジュールの属性を、逐次実行単位の分類規則に反映させることを示す概念であり、例えば、関数の分類規則により『着目関数』とされている関数にデバッグ対象のプログラムのあるスレッドの実行が移ったスレッドを『着目スレッド』と、該関数の分類規則により『非着目関数』とされている関数に実行が移ったスレッドを『非着目スレッド』と、自動的に分類するように論理設計されたプログラムや回路等によって実現される。

【0108】「制御指示群」とは、デバッグ装置に与えるべき一または複数の制御指示を時系列的に羅列した列である。

【0109】「動作過程」とは、制御指示群を与えられたデバッグ装置が、それに従ってデバッグ対象のプログラムを制御した際の、デバッグ対象のプログラムの状態を時系列的に羅列した列、及び、この際与えた制御指示群、及び、例えばマルチスレッドプログラムの場合には、スレッドの分類といった情報を総て含む概念である。

【0110】「処理命令を呼出す論理軌跡」とは、処理命令が呼出された論理的な軌跡を表す情報を示す概念であり、例えば、関数呼び出しのスタックを指す。

【0111】以上の各請求項に記載された本発明によれば次のような効果が得られる。

【0112】請求項1及び25記載の本発明では、デバッグ対象のプログラムに含まれる逐次実行単位のデバッグにあたり、特定の逐次実行単位のみを対象として制御するので、逐次実行単位間の実行のタイミングを任意に定めることが可能となる。

【0113】請求項2及び26記載の本発明では、特定動作手段がプログラムに含まれる逐次実行単位の内、特定の逐次実行単位のみについて実行を停止させるように制御するので、逐次実行単位間の実行の再開又は停止のタイミングを任意に定めることが可能となる。

【0114】請求項3及び27記載の本発明では、請求項1及び2記載の発明の作用に加えて、プログラムに含まれる各逐次実行単位を被制御形態に応じて分類し、この分類情報に適応的にデバッグ装置は制御を行うので、並列プログラムのデバッグ効率を高めることが可能となる。

【0115】請求項4及び28記載の本発明では、請求項3記載の発明の作用に加えて、プログラムに含まれる各逐次実行単位の分類を分類のための規則に基づいて行い、この分類規則をデバッグ作業者が任意に更新できるので、並列プログラムの特定逐次実行単位について制御

させる動作がより機動的になる。

【0116】請求項5及び29記載の本発明では、請求項3及び4記載の発明の作用に加えて、分類情報として逐次実行単位の被制御形態に関する情報を用いるので、より効率のよいデバッグを行うことが可能になる。

【0117】請求項6及び30記載の本発明では、請求項3及び4記載の発明の作用に加えて、分類情報として逐次実行単位の着目形態に関する情報を用いるので、デバッグ制御において着目性に適応させた制御を行うことにより、より効率のよいデバッグを行うことが可能になる。

【0118】請求項7及び31記載の本発明では、請求項3及び4記載の発明の作用に加えて、分類情報として逐次実行単位の表示形態に関する情報を用いるので、デバッグ制御において表示性に適応させた制御を行うことにより、より効率のよいデバッグを行うことが可能になる。

【0119】請求項8及び32記載の本発明では、デバッグを制御する指示として、複数の逐次実行単位のうち特定の逐次実行単位のみを対象とする特定制御指示と、デバッグ対象のプログラム全体を対象とする一般制御指示とを含んで構成されるので、デバッグ制御を細かく指示することができ、より効率の良いデバッグを行うことが可能になる。

【0120】請求項9及び33記載の本発明では、デバッグ対象のプログラムを構成する複数の逐次実行単位のうち特定の逐次実行単位とを対象としてそれぞれの分類に適応的にデバッグ制御がされるので、逐次実行単位の分類を変えればデバッグ時の挙動を変えることができ、従ってより効率の良いデバッグを行うことが可能になる。

【0121】請求項10及び34記載の本発明では、一般制御指示が入力された場合、デバッグ対象のプログラムの構成要素のうち実行または停止中の排他的逐次実行単位及び並立的逐次実行単位のみについて実行または再開させるように動作するので、プログラム全体に対する制御指示を行う際に、デバッグ作業者が逐次実行単位に対して1つ1つ制御指示を与えなければならない事態が回避され、デバッグ作業における手間を省くことが可能となる。

【0122】請求項11及び35記載の本発明では、特定制御指示が入力された場合、デバッグ対象のプログラムの構成要素のうち実行中の総ての並立的逐次実行単位及び排他的逐次実行単位のうち特定制御指示入力手段によって指定された逐次実行単位の実行を停止または再開するように動作するので、逐次実行単位ごとの詳細な制御指示を行うことにより、他の排他的逐次実行単位の実行を停止させる間に微妙なタイミングの設定をすることが可能となる。

【0123】請求項12及び36記載の本発明では、請

求項9記載の発明の作用に加えて、逐次実行単位の状態指標をデバッグ制御に反映させるので、デバッグ対象のプログラムを、デバッグ装置を介さないで実行する際には起こり得ないタイミングの実行が回避され、さらに効率的なデバッグが可能となる。

【0124】請求項13及び37記載の本発明では、制御指示入力手段に特定の逐次実行単位を対象として第1の単位実行指示が入力されたときに次に実行すべき命令が内在的停止命令である場合には特定の逐次実行単位を実行停止状態とし、該制御指示入力手段に第2の単位実行指示が入力されると次の命令を実行可能な状態にするように制御するので、第1の単位実行指示と第2の単位実行指示との間に他の排他的逐次実行単位を実行することにより微妙なタイミングの設定をすることが可能となる。

【0125】請求項14及び38記載の本発明では、プログラムに含まれる各モジュールを、着目属性に応じて分類し、特定の着目属性を持つモジュールに含まれる命令群総てをプログラムの実行の際に同一に扱うように制御するので、該プログラムに含まれるモジュールをその被着目度に応じて適切に取り扱うことが可能となる。

【0126】請求項15及び39記載の本発明では、請求項14記載の発明の作用に加えて、分類手段が、プログラムに含まれる各モジュールに対する分類規則を用いて前記各モジュールを分類する手段と、変更指示に基づいてその分類を更新する手段とを具備するので、そのモジュールの属性をデバッグ作業者が変更することが可能となり、より効率的なデバッグが可能となる。

【0127】請求項16及び40記載の本発明では、着目しないモジュールで定義された命令群、例えば関数、メソッド、手続き等の中に入り込むような『STEP IN』命令は、当該命令群全体をひとかたまりに実行してしまう『STEP OVER』命令と解釈するので、着目の対象でないモジュールに対する容易な実行制御が可能となる。

【0128】請求項17及び41記載の本発明では、プログラムに含まれる各モジュールを分類し、このモジュールの分類を考慮して逐次実行単位を分類し、この各逐次実行単位の分類に適応させてデバッグ制御を行うので、請求項1及び2に係る本願発明の目的とするところ及び請求項15に係る本願発明の目的とするところを同時に達成すること、即ち、着目の対象でないモジュールに対する容易な実行制御が可能となると同時に、モジュールの着目属性を逐次実行単位の分類に反映させた上で逐次実行単位間の実行のタイミングを任意に定めることが可能となる。

【0129】請求項18及び42記載の本発明では、デバッグログ手段に記録されたデバッグ情報を用いることにより、デバッグ装置上でデバッグ実行された場合の各逐次実行単位間の動作のタイミングと同じ、又は異なる

10

20

30

40

50

タイミングで該各逐次実行単位を動作させる動作過程を生成するので、例えばデバッグ作業者に提示することでプログラムに含まれる問題発見が容易になる。

【0130】請求項19及び43記載の本発明では、請求項18記載の発明の作用に加えてさらに、請求項18記載の発明によって生成された動作過程を自動実行し、その結果とデバッグ情報とを比較する手段を備えるので、効率的な自動テストが可能となる。

【0131】請求項20及び44記載の本発明では、デバッグ対象のプログラムに含まれる並列して実行される複数の逐次実行単位を属性に応じて分類し、該逐次実行単位ごとに設けられたウインドウに特定の属性を持つ逐次実行単位の動作状態を表示し、該ウインドウごとに、制御指示が入力されるように構成するので、並列プログラムのデバッグの動作状態を並列的に可視化することが可能となり、より効率的なデバッグが可能となる。

【0132】請求項21及び45記載の本発明では、請求項20記載の発明の作用に加えて、デバッグ対象のプログラム実行中の逐次実行単位の増減及び逐次実行単位の分類の変化に対応させてウインドウが増減されるので、逐次実行単位の状態をより反映させたデバッグを行うことができ、より効率的なデバッグが可能となる。

【0133】請求項22及び46記載の本発明では、デバッグ対象のプログラム全体に対する第1の制御指示を入力するための手段と、デバッグ対象のプログラムに含まれる並列して実行される複数の逐次実行単位を分類し、該逐次実行単位ごとに設けられたウインドウに、特定の属性の対応する逐次実行単位の動作状態と、該ウインドウに対応する逐次実行単位のみを対象とする第2の制御指示を入力するための手段とを併有させることにより、より効率的なデバッグが可能となる。

【0134】請求項23及び47記載の本発明では、逐次実行単位ごとのウインドウ画面内で逐次実行単位の実行状況に適応させた前記第2の制御指示を入力する手段のための表示を行うので、効率のよいデバッグが可能となる。

【0135】請求項24及び48記載の本発明では、デバッグ対象のプログラムに含まれる並列して実行される複数の逐次実行単位のうち特定の逐次実行単位ごとに可視的に設けられたウインドウごとに、該特定の逐次実行単位が実行中のモジュールもしくは命令群を表示させ、更に、そのウインドウに前記関数を呼出す論理軌跡を表示するので、並列プログラムのデバッグの動作状態を並列的に可視化し、デバッグ対象のプログラムの働きを理解する助けとなる。

【0136】以上から、本発明の構成によれば、例えば複数のスレッドの実行を、デバッグ作業者の希望する順序で実行制御することが可能となり、従ってデバッグ作業者は、タイミングに起因するバグの再現を常に行うことができるようになる。

【0137】

【発明の実施の形態】以下、図面を参照しながら本発明の実施の形態について説明する。

【0138】図1は、本発明に係るデバッグ装置の用いられる環境を示す概念図である。同図(a)に示すように、デバッグ装置1をデバッグ作業員2が用いることにより、ターゲットプログラム(デバッグ対象のプログラム)3のデバッグを行う場合を想定するが、同図(b)に示すように、マシンA内のデバッグ装置1がデバッグ作業員2の指示に基づいて、マシンAにリモート接続されたマシンB内のターゲットプログラム3のデバッグをリモートコントロールによって制御する環境を想定することも可能である。なおここでは、ターゲットプログラム3はマルチスレッドプログラムであるとする。

【0139】(第1の実施形態)図2は本発明の一実施形態に係るマルチスレッドプログラム3のデバッグ装置1の構成を示す概念図である。同図に示すように、デバッグ装置1は、各種制御を行うCPU101、入力装置103、出力装置105、記憶装置107、ターゲットプログラムを読み込むターゲットプログラム読み込み部109、読み込んだプログラムをデバッグのために実行するターゲットプログラム実行部111、実行中のプログラムを停止するターゲットプログラム停止部113、排他実行スレッドについて所定の実行処理を行う排他的スレッド実行機構115、スレッドの優先度により実行の制御を行うスケジューラ部117、ターゲットプログラムの実行状態及び挙動を監視するターゲットプログラム監視部119、スレッドを(後述する)各種属性によって分類するスレッド分類部121、かかるスレッドのデフォルトの分類を定めるための規則を格納するスレッド分類規則格納部123、デバッグ制御のためのコマンドを可視的に表示するデバッグコマンド表示部125、デバッグ作業員からのデバッグコマンドが入力されるデバッグコマンド入力部127、プログラム中の特定スレッドを表示するスレッド表示部129、スレッド表示の有無についてのデフォルトの規則を格納するスレッド表示規則格納部131及びプログラムあるいはスレッドの実行が一時停止されるポイントを設定するブレークポイント設定部133がバス135によって接続されて構成される。

【0140】図3は、本発明の一実施形態に係るマルチスレッドプログラム3のデバッグ装置1の構成を、デバッグ作業員2及びターゲットプログラム3との関係において示した概念図である。同図に示すような接続形態によって本実施形態に係るデバッグ装置は実現される。

【0141】図4は、デバッグコマンド入力部127の受け付けるデバッグコマンドの種別を表した図である。同図4に示すように、デバッグコマンドは、プログラム全体を対象とする一般デバッグコマンドと、特定のスレッドのみを対象とするスレッド別デバッグコマンドの二

種類に大別される。

【0142】一般デバッグコマンドには、従来のデバッグ装置と同様に、ターゲットプログラムの指定・読み込みコマンド、デバッグの開始・終了コマンド、総ての停止スレッドの実行を再開させるコマンド、総ての実行中のスレッドを停止させるコマンド等が含まれる。尚、停止状態にあるスレッドは、デバッグ装置からの指示に依らず、ターゲットプログラム自身の実行過程で停止状態になっているスレッドと、ブレークポイントに到達した結果や、デバッグ装置からの停止コマンドの結果によって、停止状態になっているスレッドの二者に大別される。本願発明では、区別を明確にするため、前者を「待機スレッド」、後者を単に「停止スレッド」と呼ぶ。

【0143】スレッド別デバッグコマンドには、停止スレッドの実行を再開するコマンド、実行中のスレッドを停止させるコマンド、各種ステップコマンド等が含まれる。後に述べるように、これらのスレッド別デバッグコマンドは、デバッグ作業者の指定した特定のスレッドに対して個別に指示される。又、ブレークポイントの設定をするコマンドも、一般デバッグコマンドと、スレッド別デバッグコマンドの二種類がある。前者によって設定されたブレークポイントには、不特定のスレッドが反応するのに対し、後者によって設定されたブレークポイントには、対象となる（一、又は、複数の）スレッドのみが反応する。

【0144】次に、スレッド分類部121及びスレッド表示部129等の働きを、図5を用いて説明する。デバッグ実行中の任意の時点で、本実施形態に係るデバッグ装置1のスレッド分類部121は、存在する各スレッドを、着目スレッド、非着目スレッドのいずれかに分類する。デバッグ作業者は、着目スレッドに分類された複数のスレッドに対して、その振る舞いや状態に着目する。又、スレッド分類部121は、各スレッドを、デバッグ装置1がどのように実行・停止させるかに応じて、排他実行スレッド、同時実行スレッド、継続実行スレッドのいずれかに分類する。ターゲットプログラム3の実行中に、新しいスレッドが生成されたときには、スレッド分類部121は、スレッド分類規則格納部123に格納されるスレッド分類規則を用いて、デフォルトの分類を決定する。またスレッド分類部121は、後にデバッグ作業者からの分類の指示によって、スレッドの分類を変更することもできる。

【0145】着目スレッドの状態は、一般に、出力装置105上に表示され続けることが望ましいが、着目スレッドの総数が多くなると、その総てを表示し続けると、逆に煩雑になる場合も有り得る。そこで、本発明では、スレッド表示部129によって、各スレッドを、表示の対象となる表示スレッド、又は、表示の対象とならない非表示スレッドに分類する。このとき、スレッド表示部129は、スレッド表示規則格納部131に格納される

スレッド表示規則を用いて、デフォルトの表示方法を決定する。また後に、デバッグ作業者からの表示方法の指示によって、表示方法を変更することもできる。デバッグコマンド表示部125は、出力装置105上に、一般デバッグコマンドの他、表示スレッドに対応するスレッド別デバッグコマンドを表示する。

【0146】スレッド分類規則の例としては、
・メインスレッド（ターゲットプログラムの入口となるスレッド）を着目スレッドとする。

【0147】・ブレークポイントで停止したスレッドを着目スレッドとする。

【0148】・ある親スレッドから生成された子スレッドは、親スレッドの分類を継承する。

【0149】・着目スレッドは排他実行スレッドとする。

【0150】・システムスレッドは、非着目スレッド、かつ、継続実行スレッドとする。

【0151】・システムスレッド以外の非着目スレッドは、同時実行スレッドとする。

【0152】・非表示スレッドは非着目スレッドとする。

【0153】等が挙げられる。また、デバッグ作業者がスレッド分類規則を変更することや、追加すること、一部のみ採用すること等を可能にしても良い。スレッド表示規則の例としては、

・着目スレッドは表示スレッドとする。

【0154】・スレッドが消滅すれば表示を終了する。

【0155】・ある親スレッドから生成された子スレッドは、親スレッドの表示方法を継承する。

【0156】・スレッドの優先順位に応じて色分けや表示順序を決定する。

【0157】等が挙げられる。スレッド分類規則と同様、デバッグ作業者がスレッド表示規則を変更することや、追加すること、一部のみ採用すること等を可能にしても良い。

【0158】次に、一般デバッグコマンド、スレッド別デバッグコマンドを与えられた排他実行スレッド、同時実行スレッド、継続実行スレッドの振る舞いを説明する。

【0159】図6は、一般デバッグコマンド、スレッド別デバッグコマンドを与えられた排他実行スレッド、同時実行スレッド、継続実行スレッド別の振る舞いの対応関係の一例を示す概念図である。

【0160】一般デバッグコマンドによって実行を再開する場合には、全停止スレッドの実行再開コマンドが与えられた場合等がある。又、一般デバッグコマンドによって停止を行う場合には、全実行中スレッドの停止コマンドが与えられた場合や、一般デバッグコマンドで設定されたブレークポイントにあるスレッドが到達した場合等がある。スレッド別デバッグコマンドによって実行を

再開する場合には、停止スレッドに対して実行再開コマンドが与えられた場合や、各種ステップコマンドが停止スレッドに与えられた場合等がある。又、スレッド別デバッグコマンドによって停止を行う場合には、実行中スレッドに対して停止コマンドが与えられた場合や、各種ステップコマンドを与えられたスレッドが停止条件（例えば、『Step In』コマンドならば、一行の実行が終了する）を満たした場合、スレッド別ブレークポイントに対象のスレッドが到達した場合等がある。ここに挙げた再開・停止の条件は例示であって、一般・スレッド別デバッグコマンドによって実行を再開・停止する場合としては、他の場合もあり得る。

【0161】図6に示すように、一般デバッグコマンドによって実行を再開する場合、排他実行スレッド及び同時実行スレッドのうち停止状態にあるスレッドが実行を再開する。一般デバッグコマンドによって実行を停止する場合、排他実行スレッド、同時実行スレッドのうち実行状態にあるスレッドが総て停止する。一方、総ての継続実行スレッドは、常にそのときの停止状態、実行状態を維持する。なお、ここに説明した一般デバッグコマンドによる実行の再開・停止動作は例示であって、他の動作、例えば、一般デバッグコマンドによって実行を停止する場合に、実行状態にある総ての継続実行スレッドの実行を停止させるようにしてもよい。

【0162】スレッド別デバッグコマンドによって実行を再開する場合、排他実行スレッドは、対象となるスレッドのみ、停止状態ならば実行を再開する。その他総ての排他実行スレッドは、その時の状態を維持する。一方、同時実行スレッドは、総ての停止状態にあるスレッドが、実行を再開する。スレッド別デバッグコマンドによって実行を停止する場合、排他実行スレッドは、対象となるスレッドのみ、実行状態ならば停止する。その他総ての排他実行スレッドは、その時の状態を維持する。一方、同時実行スレッドの実行状態にあるスレッドは総て停止する。継続実行スレッドは、常にそのときの停止状態、実行状態を維持する。

【0163】次に、以上のように構成される本デバッグ装置の動作を説明する。

【0164】図7は、デバッグ実行中に、あるスレッドが生成された時の、スレッド分類部121及びスレッド表示部129の処理の動作を説明するためのフローチャートである。

【0165】同図に示すように、ターゲットプログラムの実行過程において、新しいスレッドが生成された場合を考える（ステップ701）。

【0166】まず、上述したスレッド分類規則を用いてデフォルトの分類を定める（ステップ702）とともに、同様にデフォルトの表示方法を定める（ステップ703）。

【0167】ここでこの分類により、表示対象のスレ

ッドかどうかを検証し（ステップ704）、表示スレッドならば出力装置105上に表示する（ステップ705）。表示スレッドでなければ、既に表示されている場合には出力装置105上の表示を終了させる（ステップ706）。

【0168】次にデバッグ作業からの分類の指示が与えられた場合（ステップ707）にはスレッドの分類を更新し（ステップ708）、デバッグ作業からの分類の指示が与えられていない場合には、デバッグ作業からの表示方法の指示が与えられているかを確認する（ステップ709）。デバッグ作業からの表示方法の指示が与えられているならば、表示方法を更新し（ステップ710）、スレッド分類規則を用いてデフォルトの分類を定め（ステップ702）た後にステップ704にかえる。ステップ709で、デバッグ作業からの表示方法の指示が与えられていなければ、スレッドが消滅したかを確認（ステップ711）、消滅していなければ、ステップ707に戻る。

【0169】スレッドが消滅した場合には（ステップ711）、着目スレッドでなければ（ステップ712）そのまま、着目スレッドならば（ステップ712）デバッグ作業に該消滅を連絡した（ステップ713）上で表示スレッドかどうかを確認（ステップ714）、表示を終了させて（ステップ715）、一連の処理を終了する。

【0170】ここで表示スレッドでなければ（ステップ714）、そのまま処理を終了する。

【0171】次に、スレッド別デバッグコマンドである『Step In』をデバッグ作業が指示した場合の動作を、図8のフローチャートを用いて説明する。

【0172】同図に示すように、まず、スレッド別デバッグコマンド『Step In』が入力される（ステップ801）。デバッグコマンド入力部127は、対象になるスレッドTを決定する（ステップ802）。同時実行スレッドのうち停止状態にあるスレッド総ての実行を再開し（ステップ803）、スレッドTの実行を再開する（ステップ804）。

【0173】スレッドTの実行中ブレークポイントに達したスレッドがある場合（ステップ805）には、既知の技術であるブレークポイント処理（ステップ806）を行う。ブレークポイントに達したスレッドがない場合（ステップ805）には、スレッドTが1行の実行を終えたかを確認する（ステップ807）。

【0174】スレッドTが1行の実行を終えていない場合にはステップ805に戻り、スレッドTが1行の実行を終えた場合には、スレッドTの実行を停止する（ステップ808）。

【0175】最後に同時実行スレッドのうち実行状態にあるスレッド総ての実行を停止して（ステップ809）終了する。

【0176】ここで述べた、スレッドの実行形態による分類に対する、スレッドの実行及び停止は、ターゲットプログラム実行部111、ターゲットプログラム停止部113、排他的スレッド実行機構115、ターゲットプログラム監視部119、スレッド分類部121によって実現される。具体的な実現方法は、OSに依存している。この実現方法の一例として、OSの提供するデバッグ機能が、ある特定のスレッドの実行を再開すると、その他の総てのスレッドの実行も再開し、あるスレッドの実行が停止すると、その他総てのスレッドの実行も停止する10という場合に、図8に示した『Step In』を実現するときの処理の流れを次に説明する。

【0177】図9は、『Step In』を実現するにあたっての排他実行スレッドの実行の再開及び停止の動作を示すフローチャートである。

【0178】同図に示すように、排他実行スレッドの実行の再開にあたっては、まず、スレッドT以外の停止状態にある排他実行スレッドを集合Sに記憶する(ステップ901)。具体的には、集合Sの内容を記憶装置107内に記憶する。

【0179】次に、集合Sに含まれるスレッドに『OS_suspend』を発行する(ステップ902)。

【0180】スレッドTが停止すべき位置に、スレッドTのみが反応するブレークポイントを設定し(ステップ903)、スレッドTの実行を再開する(ステップ904)。排他実行スレッドの実行の停止にあたっては、まず、スレッドTの実行を停止し(ステップ912)、スレッドTに対して『OS_suspend』を発行する(ステップ914)。

【0181】ここで任意のスレッドの実行を再開する(ステップ915)。これにより、『OS_suspend』を発行されていない総てのスレッドの実行が再開される。

【0182】次に、集合Sに含まれるスレッドに『OS_resume』を発行し(ステップ916)、スレッドTに対しても『OS_resume』を発行して(ステップ917)処理を終了する。

【0183】なお、同図において、『OS_suspend』及び『OS_resume』は、例にあげたOSの提供するデバッグの命令で、この『OS_suspend』を発行されたスレッドは、その他のスレッドの実行が再開しても、実行を再開せず、また、『OS_resume』は、先に発行された『OS_suspend』命令を取り消す命令であると仮定している。ここでは、OSによってスレッド別ブレークポイントが提供されていることを仮定したが、スレッド別ブレークポイントが提供されていない場合には、デバッグ装置側でスレッド別ブレークポイントをエミュレートする必要がある。

【0184】次に、スレッド別デバッグコマンドの『S

tep In』を例にして、本発明の一実施形態に係るデバッグ装置の付加的機能を説明する。

【0185】デバッグのある時点で、ある排他実行スレッドが、ターゲットプログラム中の『sleep』命令(実行することにより、そのスレッドは一定時間待機状態になり、一定時間経過後、自発的に実行状態になる。このような待機状態を休眠状態とよぶ)を次に実行する状態で停止しているとする。デバッグ作業者がこの状態で、そのスレッドにスレッド別デバッグコマンド『Step In』を発行した場合、そのスレッドに、実際に『sleep』命令を実行させ、一定時間休眠状態にすることもできるが、本発明においては、更に、図10に示す動作により、あるスレッドが『sleep』状態にある間に、その他のスレッドが動くタイミングを、デバッグ作業者が自由に選んでデバッグすることを可能とする。

【0186】図10は、かかる機能を実現するための動作を説明するためのフローチャートである。

【0187】同図に示すように、まず、排他実行スレッドTが、次に『sleep』命令を実行する直前で停止状態にある(ステップ1001)とする。

【0188】スレッドTに対して『Step In』コマンドが発行されると(ステップ1002)、スレッドTは停止状態になる(ステップ1003)。即ち、『Step In』コマンドを発行された排他実行スレッドが、『sleep』命令を実行しようとする、直ちに、そのスレッドを停止状態にする。そしてTの第1の『Step In』コマンドは終了する(ステップ1004)。この際、デバッグ作業者にに対しそのスレッドが『sleep』状態に入ったことを示しても良い。

【0189】デバッグ作業者は、この状態において、その他の排他実行スレッドを任意に制御することができる。sleep命令により停止状態になったスレッドに対し、デバッグ作業者が第2の『Step In』コマンドを発行する(ステップ1005)ことにより、このスレッドは、『sleep』命令を抜け、次の命令を実行できる状態になり(ステップ1006)、第2の『Step In』コマンドは終了する(ステップ1007)。

【0190】なお同図中、画面のイメージ1001a、1003a及び1006aは、それぞれステップ1001、1003及び1006におけるスレッドTが次に実行する位置を指標する様子を表した画面図である。

【0191】この機能によって、あるスレッドが『sleep』状態にある間に、その他のスレッドが動くタイミングを、デバッグ作業者は自由に選んでデバッグすることが可能となる。同様に、例えば、ターゲットプログラム中の『suspend』命令(実行することによりそのスレッドは待機状態になる。待機状態から抜けるためには、その他のスレッドから『resume』命令

37

を受ける必要がある)を『Step In』コマンドで実行した場合の動作を図11のフローチャートを用いて説明する。

【0192】同図に示すように、まず、排他実行スレッドTが、次に『suspend』命令を実行する直前で停止状態にある(ステップ1101)とする。

【0193】スレッドTに対して『Step In』コマンドが発行されると(ステップ1102)、スレッドTは待機状態になる(ステップ1103)。即ち、『Step In』コマンドを発行された排他実行スレッドが、『suspend』命令を実行しようとする、直ちに、デバッグ装置はそのスレッドを待機状態にする。この際、デバッグ作業員に対しそのスレッドが『suspend』状態に入ったことを示しても良い。

【0194】上記の『suspend』命令を実行したスレッドが、待機状態に入った時点で、デバッグ装置は『Step In』コマンドの実行を終える(ステップ1104)。

【0195】デバッグ作業員はこの状態において、その他の排他実行スレッドを任意に制御することができる。そして、その過程において、他のスレッドが待機状態にある当該スレッドに『resume』命令を発行すると(ステップ1105)、発行されたスレッドは、待機状態から停止状態に移移する(ステップ1106)。

【0196】更に、このスレッドにデバッグ作業員が第2の『Step In』コマンドを発行して初めて(ステップ1107)、このスレッドは、『suspend』命令の次の命令を実行できる状態になり(ステップ1108)、第2の『Step In』コマンドは終了する(ステップ1109)。又、ステップ1106、ステップ1107は省略するという実装も考えられる。

【0197】なお同図中、画面のイメージ1101a、1103a、1106a及び1108aは、それぞれステップ1101、1103、1106及び1108におけるカレントスレッドの位置を指標する様子を表した画面図である。

【0198】以上の手法により、デバッグの任意の時点で、デバッグ作業員が複数のスレッドの間の実行のタイミングによるプログラムの挙動を確認することを意図する場合、これらのスレッドを着目スレッド(従って、デフォルトで排他実行スレッドとなる)と指定すれば、これらのスレッドをデバッグ作業員の希望する任意の順序で容易にステップ実行することが可能となる。

【0199】なお、スレッド間に優先度が設けられている場合には、上述の手法のみでは、デバッグ装置を介しない本来のターゲットプログラムの実行では、起こり得ないタイミングを許すことが生じ得る。例えば、スレッドt1の優先度がスレッドt2の優先度より高い場合を考える。実際には、スレッドt1の実行中には、スレッドt2の実行は行われない。しかし、上述の手法のみで

38

は、例えば、次のような問題が生ずる。

【0200】(1)スレッドt1が排他実行スレッド、スレッドt2が同時実行スレッドと指定されており、どちらも停止状態である場合、もしも、スレッドt1以外の停止した排他実行スレッドの実行を再開すると、スレッドt1は停止しているにもかかわらず、スレッドt2の実行は再開されてしまう。

【0201】(2)スレッドt1、t2が共に、排他実行スレッドに指定されている場合、デバッグ作業員は、スレッドt2を、スレッドt1を停止させたまま、実行させることができる。

【0202】これらの実際には起こり得ないタイミングの実行を避ける為、本発明では、スケジューラ部117が、本来のスレッドの挙動をエミュレートする。例えば、上記(1)の場合には、ターゲットプログラム実行部111は、スレッドt2に対して、実行の再開を指示するが、スレッドt2よりも優先度の高いスレッドt1が停止しているために、最終的に、スケジューラ部117がスレッドt2の実行を行わない。

【0203】また上記(2)の場合には、デバッグ作業員が、スレッドt2を実行させようとしたときに、スケジューラ部117が、実際には起らないタイミングでの実行を行おうとしている旨をデバッグ作業員に伝えることができる。さらに(2)に関しては、ターゲットプログラム監視部119が、デバッグコマンド表示部125にt2を実行するコマンドを表示させないようにすることもできる。

【0204】次に、本発明の一実施形態に係るデバッグ装置のGUI上の動作を説明する。図12は、本発明の一実施形態に係るデバッグ装置をGUIを用いて実現した場合の画面の一例を示した図である。

【0205】同図において、スレッドトレースウインドウ1201は、従来のデバッグ装置のプログラム表示ウインドウとコントロールパネルとを一つにまとめたものである。但し、従来デバッグ装置のように、ターゲットプログラムに対してコントロールパネルが一つだけ存在するのではなく、従来のコントロールパネルに相当する本発明のスレッド別コントロール部は、排他実行スレッドかつ表示スレッドである各スレッドに対して一つずつ存在する。これにより、スレッド別デバッグコマンドを各スレッドに個別に与えることが容易になる。

【0206】スレッドトレースウインドウ1201には、排他実行・表示スレッドに対応するスレッド別コントロール部が含まれる。同図の例では、三つの排他実行・表示スレッドmain、th1、th2が存在する為、それぞれに対する個別スレッドコントロール部1202、1203及び1204がスレッドトレースウインドウ1201に含まれている。

【0207】スレッド別コントロール部1202、1203及び1204は、ターゲットプログラム3実行中の

スレッドの生成及び消滅、並びにスレッド分類の変化等に連動して自動的に増減させることができる。

【0208】スレッド別コントロール部1202、1203及び1204は、スレッドトレースウインドウ1201内にまとめて表示するのではなく、スレッドごとに異なるウインドウに表示するという実現例も考えられる。

【0209】尚、一般デバッグコマンドを与えるためには、一般コントロールパネルを用意する方法や、ウインドウシステムのメニューを用いる方法等が考えられる。 10

【0210】図13は、このような一般コントロールパネルの一例を示す図である。

【0211】同図に示す一般コントロールパネル1300は、スレッドボタン1301、ブレークポイント設定ボタン1302、一般デバッグコマンド指示部1303が含まれる。

【0212】スレッドボタン1301を押すと、(後述するように)ウインドウに着目スレッド及び非着目スレッドの一覧が表示される。このウインドウや、メニューバーを用いて、デバッグ作業者はスレッドの分類の指示や、表示方法の指示をスレッド分類部121及びスレッド表示部129に与えることができる。このウインドウには、各スレッドの優先順位や現在の状態等の詳細な情報を表示させることもできる。 20

【0213】ブレークポイント設定ボタン1302を押すと、全てのスレッドに反応するブレークポイント等を設定することができる。

【0214】一般デバッグコマンド指示部1303は、一般デバッグコマンドを列挙したボタン群よりなる。同図の例では、全停止スレッドの実行を再開させるContボタン1303a、全実行中スレッドを停止させるStopボタン1303b、デバッグを終了させるExitボタン1303cが含まれている。 30

【0215】一般デバッグコマンドには、ここに挙げたものの他に、デバッグ対象となるターゲットプログラム3の指定、デバッグの開始コマンド等も含まれる。これらのコマンドは比較的使用頻度が低いので、一般コントロールパネル1300には含まれておらず、メニューバーを通して指示される。

【0216】次に、スレッドトレースウインドウ1201における、各スレッド別コントロール部1202、1203及び1204の働きを図14を用いて説明する。 40

【0217】図14は、図12におけるスレッド別コントロール部1203の部分を取り出して表示した図である。

【0218】同図に示すように、スレッド別コントロール部1203には、非着目化ボタン1401、スレッド状態表示アイコン1402、スレッド名表示部1403、スタックフレーム表示選択部1404、スレッド別デバッグコマンド指示部1405、ソース表示部140 50

6、位置ポインタ1407、ブレークポイントポインタ1408等が含まれる。

【0219】非着目化ボタン1401を押すことにより、スレッド分類部121は対応するスレッドを非着目スレッドにする。これは、図7のフローチャート上ではデバッグ作業からの表示方法の指示(ステップ707)によって、スレッドの表示方法を更新する部分(ステップ710)に相当する。続いて、ステップ702でスレッド分類規則を用いて、指定されたスレッドを同時実行スレッドにする。この後、ステップ704及び706で指定されたスレッドを非表示スレッドにする。

【0220】スレッド状態表示アイコン1402は、対応するスレッドの状態(実行状態、停止状態、自発的停止状態等)を可視的に表示する。

【0221】スレッド名表示部1403は、対応する着目スレッドの名前を表示する。スレッド名をここでデバッグ作業に変更させることもできる。

【0222】スタックフレーム表示選択部1404は、ソース表示部1406で表示する関数呼び出しスタックフレームをデバッグ作業者に表示選択させる部分である。スタックフレーム表示選択部1404の中の、プルダウンボタン1404aをクリックした場合の画面の一例を図15に示す。

【0223】同図に示す例では、スレッドth1は、関数でrun()の実行中に、関数sub1()を呼び出し、その実行中に、関数g()を呼び出し、更にその内部で停止状態になっている。デバッグ作業者は、ソース表示部1406に表示するソースプログラムを、関数sub1()を定義したものから、関数g()を定義したものに換えようとしている。

【0224】スタックフレーム表示選択部1404には、関数の定義されているファイル名等を合わせて表示することもできる。

【0225】スレッド別デバッグコマンド指示部1405は、対応するスレッドに対するスレッド別デバッグコマンドを列挙したボタン群より成る。図14に示した例では、停止スレッドの実行を再開させる『Cont』ボタン1405aや、ステップ実行させる『Step In』『Step Over』『Step Out』ボタン1405b、1405c及び1405dが含まれている。

【0226】スレッド別デバッグコマンド指示部1405には、更に、対応するスレッドのローカル状態(例えば、スタックフレーム表示選択部1401に示されたスタックフレーム中で定義された変数の現在値等)を表示させるボタン等も含めることができる。さらに、スレッド別デバッグコマンド指示部1405に含まれるボタンを、対応するスレッドの現在の状態によって可変にすることもできる。例えば、図12におけるスレッド別コントロール部1202のスレッド別デバッグコマンド指示

41

部には、対応するスレッドmainが実行中なので、その実行を停止させる『Stop』ボタン1202aが含まれている。

【0227】また、図12におけるスレッド別コントロール部1204では、対応するスレッドth2がsuspend命令による待機状態になっているため、スレッド別デバッグコマンド指示部1405に含まれるボタンが総て押せない状態になっている。

【0228】ソース表示部1406では、スタックフレーム表示選択部1404に示されたスタックフレームの関数が定義されたファイルの一部を表示する。このファイルの適当な行をデバッグ作業者がクリックする等して指示することにより、対応するスレッドのみが反応するスレッド別ブレークポイントを設定することも可能である。

【0229】位置ポインタ1407は、対応するスレッドの現在の行を示す。ブレークポイントポインタ1408は、そのポインタの示す位置にブレークポイントが置かれていることを示す。ポインタの形状によって、ブレークポイントの種類（例えば、全ての着目スレッドに反応するブレークポイント、スレッド別ブレークポイント、全てのスレッドに反応するブレークポイント等）をデバッグ作業者に可視的に示すこともできる。

【0230】図16は、図13に示した一般コントロールパネル1300においてスレッドボタン1301を押すことにより表示されるスレッドの一覧画面の一例を示す図である。

【0231】同図に示す例では、着目スレッド（『*』のついたスレッド）、非着目スレッド（『*』のついていないスレッド）の一覧が表示される。

【0232】次に、デバッグコマンドに対する排他実行スレッド及び同時実行スレッドのGUIに関連する動きを、例を用いて説明する。

【0233】今、デバッグ実行中のある時点で、ターゲットプログラム3中に、6つのスレッドmain、th0、th1、th2、th2及びth4が存在し、そのうち、3つのスレッドmain、th1及びth2をデバッグ作業者が着目スレッドとして指定しているとす。これらのスレッドは、デフォルトで排他実行スレッドに分類されている。またスレッドth4は、非表示スレッドではあるが、排他実行スレッドとしてデバッグ作業者に指示されているとする。その他のスレッドth0及びth3は、デフォルトで同時実行スレッドに分類されている。

【0234】この時点で、スレッドmainは実行状態（図12における1202で表示）にあり、スレッドth1はある行で停止（図12における1203で表示）しており、スレッドth2はある行で待機状態（図12における1204の状態）にあり、スレッドth4はある行で停止しているとする。

42

【0235】デバッグ作業者が、スレッドth1にスレッド別デバッグコマンド『StepIn』を与えた場合、スレッドth1の実行と、全ての同時実行スレッド（th0、th3）の実行が再開される。このとき、従来のデバッグ装置とは異なり、スレッドth1以外の排他実行スレッドは、現在の状態を維持する。即ち、スレッドmainは実行状態、スレッドth2は待機状態、スレッドth4は停止状態を維持する。但し、もし、スレッドmainの優先度がスレッドth4の優先度より低い場合には、実際にはスケジューラ部117が、スレッドmainの実行を行わない。

【0236】またもし、スレッドth1の優先度がスレッドth4の優先度より低い場合には、スケジューラ部117によって、実際には起こり得ないタイミングでステップ実行を行おうとしていることがデバッグ作業者に伝えられる。それにもかかわらず、デバッグ作業者が敢えてスレッドth1をステップ実行することを選んだ場合には、スレッドth1の実行が実際に行われるようにすることができる。

【0237】スレッドth1が一つの行を実行し終わると、スレッドth1及び全ての同時実行スレッドは停止状態になる。このときも、スレッドth1以外の排他実行スレッドは、現在の状態を維持する。

【0238】もしスレッドth1のステップ実行途中において、同時実行スレッドth0、th3、又は実行中の排他実行スレッドmainの内のいずれかのスレッド（例えばth3）が非着目スレッドに対しても反応するブレークポイントに到達した場合、排他実行スレッドは現在の状態を維持し、th3及びその他の同時実行スレッドth0は停止させられる。

【0239】以上説明したように本実施形態によれば、並列プログラム、例えばマルチスレッドプログラム、を対象とするデバッグ装置において、スレッドを排他実行スレッドとそうでないスレッドに分類して扱い、デバッグコマンドとしてスレッドに個別に与え得るものとスレッド全体に与え得るものとを設け、排他実行スレッドに指定されたものは同時に実行させないように動作させることにより、複数のスレッドの実行を、デバッグ作業者の希望する順序で制御することが可能となり、従ってデバッグ作業者は、タイミングに起因するバグの再現を常に行うことができるようになる。

【0240】（第2の実施形態）次に、本発明の他の実施形態を説明する。

【0241】上述した実施形態では、スレッドの分類は、スレッド分類規則によるデフォルトの分類及びデバッグ作業からの分類の指示に依っていたが、本実施形態においては、プログラムに含まれるモジュール、又は、モジュールに含まれる関数、を属性によって分類することによる、更に高度なスレッドの分類方法を示す。

【0242】図17は本実施形態に係るデバッグ装置の

構成を示すブロック図である。

【0243】同図に示すように、本デバッグ装置の構成は図3に示す構成に、関数分類部1701及び関数分類規則格納部1703を付加して成り立つ。但し図17は本デバッグ装置の構成の主要部を示したものであり、図3の構成と同様な箇所については記載を省略しているものもある。また図17において図3の構成と同等の部分については図3の場合と同一番号を付し、以下の説明においては当該共通部分については機能の説明を省略する。

【0244】関数分類部1701は、図18にその概念が示されるように、ターゲットプログラム3中で定義された任意の関数（手続きを含む）を、例えば着目関数又は非着目関数のいずれかに分類する機能を有する。この場合、デバッグ作業から特に関数分類の指示が与えられない限り、デフォルトでは関数分類規則格納部1703に格納されている関数分類規則を用いて分類が行われる。

【0245】関数分類規則の例としては、
・ターゲットプログラムで定義された関数を全て着目関数とする。

【0246】・関数名に特定の文字列を含む物のみを着目関数とする。

【0247】・オブジェクト指向プログラミング言語を対象としている場合、デバッグ作業者に指定されたクラスのクラスメソッド及びインスタンスメソッドのみを着目関数とする。

【0248】・オブジェクト指向プログラミング言語を対象としている場合、デバッグ作業者に指定されたインスタンスのインスタンスメソッドのみを着目関数とする。

【0249】等が挙げられる。デバッグ作業者は、これらの規則のいずれを採用するかを選択することや新しい規則を設けることができる。

【0250】これに応じて、スレッド分類規則には、新しく、

・着目関数に実行が移ったスレッドを着目スレッドとする。

【0251】・非着目関数に実行が移ったスレッドを非着目スレッドとする。

【0252】等の規則を加える。

【0253】更にデバッグ作業者が、例えばスレッド別デバッグコマンド『Step In』をあるスレッドに与えたとき、そのスレッドが着目関数から、非着目関数を呼び出すことになった場合には、デバッグ装置1側では、このコマンドを『Step Over』コマンドと解釈して、非着目関数内部のステップ実行を行わないようにすることもできる。

【0254】なお、以上の説明においては、モジュールとして関数を例にとり説明したが、モジュールたり得る

他の要素、例えば、クラス、メソッド、サブルーチン等を対象としても同様の技術思想を適用することが可能である。

【0255】以上説明したように、本実施形態に係るデバッグ装置によれば、プログラムに含まれる各モジュールを着目形態に応じて分類し、この着目形態に応じてデバッグ制御するので、細かいデバッグ制御が可能となる。従って、第1の実施形態の場合のスレッド分類規則と本実施形態の場合の関数分類規則とを適切に対応付けることによれば、第1の実施形態の場合にデバッグ作業にかかると手間を減らし、デバッグにおける利便性を一層増すことが可能となる。

【0256】（第3の実施形態）次に、本発明の更に別の実施形態について説明する。

【0257】本実施形態においては、ターゲットプログラム3の自動テスト実行装置の実現例を述べる。

【0258】図19は本実施形態に係る自動テスト装置の構成を示すブロック図である。

【0259】同図に示すように、本テスト装置の構成は図3に示す構成に、テストパターン生成部1901及びデバッグログ記録部1903を付加して成り立つ。但し図19は本テスト装置の構成の主要部を示したものであり、図3の構成と同様な箇所については記載を省略しているものもある。また図19において図3の構成と同等の部分については図3の場合と同一番号を付し、以下の説明においては当該共通部分については機能の説明を省略する。

【0260】デバッグログ記録部1903では、作業20の入力した各種デバッグコマンド、スレッドの分類、ターゲットプログラム3の状態等の時系列情報（シナリオ）を記録する。これに加えてデバッグログ記録部1903では更に、記録している各シナリオごとに、ターゲットプログラム3の振る舞いにバグが含まれているか否かを作業者に判定させ、その判定結果を記録することもできる。

【0261】テストパターン生成部1901では、デバッグログ記録部1903に記録された情報を基に、作業20によって実行の行われたスレッド制御のタイミングとは異なるタイミングでスレッドの動作を制御するようなテストパターンを自動生成する。

【0262】作業20の入力するデバッグコマンドに代えて、テストパターン生成部1901で作られたテストパターンをデバッグ装置1に与えることにより、テストの自動実行を行うことができる。又、作業20によって実行の行われたスレッド実行のタイミングと同じタイミングでスレッドの動作を制御すれば、作業者が過去に行った作業を再生することができる。そして、この再生の終了した後、又は、再生を途中で中断した後、作業20は、テスト作業を続行し、記録されたタイミングとは異なる新たなスレッド実行のタイミングでスレッド

10

20

30

40

50

の動作を制御して手動でテストを行うこともできる。

【0263】本実施形態に係るテスト装置を用いた場合の最大の特徴は、スレッド実行のタイミングを、テストの自動実行の都度完全に制御できる点にある。

【0264】自動テスト実行についての具体的な実現方法については、例えば同一出願人による特願平9-288200号公報に開示された技術思想を用いることが可能である。

【0265】即ち、プログラムをテスト動作させ、その実行の際のログ（実行された文とその順序、外部からの入力、外部への出力等の情報）を取り出し、この実行のログからプログラム動作として起こり得るものであってもとの動作とテストの判定が異なる可能性のある動作を発見させる。この発見をさせるために、プログラムを観察有界と捉え、観察有界なプログラムでは与えられた具体的シナリオを入出力の制約として解くことにより非決定的な動作を総て網羅できる点を利用して膨大な量のシナリオの中から対象を限定させる。

【0266】このような自動テスト装置に対して本実施形態に係る技術的思想を適用すれば、例えばデバッグログ記録部1903に記録されたデバッグ情報を上記の自動テスト装置のテスト対象として用いることにより、デバッグ装置上でデバッグ実行された場合の各処理間の動作のタイミングと異なるタイミングで該各処理を動作させる動作過程を生成させ、結果として並列プログラムの実行のタイミングを制御した上でテストの自動実行をすることが可能となる。

【0267】また自動テスト本来の、冗長なシナリオを除いた効率の良いテストを並列プログラムの実行の微妙なタイミングの制御を行いつつ実現することができる。

【0268】以上詳述したように、本実施形態のテスト装置によれば、並列プログラムの実行を作業者の希望する順序で実行することが可能となり、従って作業者は、タイミングに起因するバグの再現を常に行うことができるようになる。

【0269】従って本実施形態のテスト装置によれば、テスト、特に並列プログラム、例えばマルチスレッドのプログラムのテストにおいて、微妙なタイミングの設定を行うことが可能となる。

【0270】従って本実施形態のテスト装置によれば、効率的なテストを行うことが可能となる。

【0271】なお、本発明は、上述した実施形態には限定されず、本発明の技術思想の範囲内で様々な変形が可能である。

【0272】例えば、上述した実施形態では、本発明の技術思想を具現化する対象としてデバッグ装置およびテスト装置を例にとり説明したが、例えば同様のデバッグ機能をコンピュータに果たさせるようなプログラムを技術思想の具現化対象とすることも可能である。

【0273】また、上述した実施形態においては、並列

プログラムとしてマルチスレッドプログラムを例にとり、このプログラムをデバッグする場合について説明したが、並列プログラムとしてマルチプロセスプログラム、マルチタスクプログラム等を用いても本発明の技術思想を適用することが可能である。

【0274】また、上述した実施形態においては、デバッグ対象の並列プログラム、例えばマルチスレッドプログラムのスレッドをスレッド分類部が、着目スレッド及び非着目スレッドのいずれかに分類するとともに、同対象スレッドを排他実行スレッド、同時実行スレッド、継続実行スレッドのいずれかに分類する場合を説明した。しかしスレッド分類部はこの2種類の分類を常に同時にしなければならないものではなく、どちらか一方の分類を行うようにしても本発明の技術思想は適用できる。さらには、分類態様自体が着目・実行分類に限られるものでなく、他の分類態様を採用することによってもよい。

【0275】また、実行属性／着目属性に応じた実行・停止の対応関係を律する実行規則／着目規則は図6の場合を例として説明したが、同図以外の実行規則／着目規則を採用する場合であってももとより本発明の技術思想は適用可能である。

【0276】また上述した実施形態では、排他実行スレッドは表示スレッドとされる論理を背景としたが、他の論理を背景にする場合であっても本発明の技術思想は適用可能である。表示規則についても同様に、上述した実施形態の場合に限定されるものではない。

【0277】また上述した実施形態では、実際には起こり得ないタイミングでのプログラム実行を防止するために本来のスレッドの挙動をエミュレートするべく、スケジューラ部が参照すべき各処理（例えばスレッド）の状態指標として「優先度の高いスレッドが停止しているときは、より低い優先度に係るスレッドの実行は行わない」を例にとり説明したが、状態指標としては本来のスレッドの挙動を表すものであれば総て本発明の技術思想の適用対象となり得る。

【0278】また上述した実施形態では、モジュール（例えば、関数）の着目属性による分類を考慮して処理（例えば、スレッド）を分類させる規則として「着目関数に実行が移ったスレッドを着目スレッドとし、非着目関数に実行が移ったスレッドを非着目スレッドとする」を例にとり説明したが、当該分類規則としてはこの例の場合に限定されるものでなく、他の分類規則であってももとより本発明の技術思想は適用可能である。

【0279】また上述した実施形態では、プログラムの動作過程として各種デバッグコマンド及びターゲットプログラムの状態等の時系列情報（シナリオ）を例にとり説明したが、動作過程はここで挙げたものに限定されるものでなく、プログラムの実行手順や挙動などを含めた動作過程たる概念に対して、本発明の技術思想は適用可能である。

10

20

30

40

50

【0280】さらには上述した実施形態では、特定の処理（例えば、スレッド）ごとに設けられたウインドウに表示された処理単位（例えば、スレッド中のプログラム）を特定する手段としてプルダウンボタンをクリックする場合を例にとり説明したが、該特定手段はこの方法に限定されるものでなく、他の特定手段、例えばキーボードからの入力等を用いる場合においても、本発明の技術思想は適用可能である。

【0281】また上述した実施形態では、特定指示の入力によりウインドウ画面に表示される処理命令（例えば、関数）を呼出す論理軌跡として、当該関数が呼び出された論理的な履歴を例にとり説明したが、論理軌跡はこのような履歴に限らず、当該処理命令が実行されるためのプログラム上の手順を表現するもの一般に対して本発明の技術思想を適用することが可能である。

【0282】

【発明の効果】以上詳述したように、請求項1及び25記載の本発明では、デバッグ対象のプログラムに含まれる逐次実行単位のデバッグにあたり、特定の逐次実行単位のみを対象として制御するので、逐次実行単位間の実行のタイミングを任意に定めることが可能となる。

【0283】請求項2及び26記載の本発明では、特定動作手段がプログラムに含まれる逐次実行単位の内、特定の逐次実行単位のみについて実行を停止させるように制御するので、逐次実行単位間の実行の再開又は停止のタイミングを任意に定めることが可能となる。

【0284】請求項3及び27記載の本発明では、請求項1及び2記載の発明の作用に加えて、プログラムに含まれる各逐次実行単位を被制御形態に応じて分類し、この分類情報に適応的にデバッグ装置は制御を行うので、並列プログラムのデバッグ効率を高めることが可能となる。

【0285】請求項4及び28記載の本発明では、請求項3記載の発明の作用に加えて、プログラムに含まれる各逐次実行単位の分類を分類のための規則に基づいて行い、この分類規則をデバッグ作業者が任意に更新できるので、並列プログラムの特定逐次実行単位について制御させる動作がより機動的になる。

【0286】請求項5及び29記載の本発明では、請求項3及び4記載の発明の作用に加えて、分類情報として逐次実行単位の被制御形態に関する情報を用いるので、より効率のよいデバッグを行うことが可能になる。

【0287】請求項6及び30記載の本発明では、請求項3及び4記載の発明の作用に加えて、分類情報として逐次実行単位の着目形態に関する情報を用いるので、デバッグ制御において着目性に適応させた制御を行うことにより、より効率のよいデバッグを行うことが可能になる。

【0288】請求項7及び31に係る本発明では、請求項3及び4記載の発明の作用に加えて、分類情報として

逐次実行単位の表示形態に関する情報を用いるので、デバッグ制御において表示性に適応させた制御を行うことにより、より効率のよいデバッグを行うことが可能になる。

【0289】請求項8及び32記載の本発明では、デバッグを制御する指示として、複数の逐次実行単位のうち特定の逐次実行単位のみを対象とする特定制御指示と、デバッグ対象のプログラム全体を対象とする一般制御指示とを含んで構成されるので、デバッグ制御を細かく指示することができ、より効率の良いデバッグを行うことが可能になる。

【0290】請求項9及び33記載の本発明では、デバッグ対象のプログラムを構成する複数の逐次実行単位のうち特定の逐次実行単位とを対象としてそれぞれの分類に適応的にデバッグ制御がされるので、逐次実行単位の分類を変えればデバッグ時の挙動を変えることができ、従ってより効率の良いデバッグを行うことが可能になる。

【0291】請求項10及び34記載の本発明では、一般制御指示が入力された場合、デバッグ対象のプログラムの構成要素のうち実行または停止中の排他的逐次実行単位及び並立的逐次実行単位のみについて実行または再開させるように動作するので、プログラム全体に対する制御指示を行う際に、デバッグ作業者が逐次実行単位に対して1つ1つ制御指示を与えなければならない事態が回避され、デバッグ作業における手間を省くことが可能となる。

【0292】請求項11及び35記載の本発明では、特定制御指示が入力された場合、デバッグ対象のプログラムの構成要素のうち実行中の総ての並立的逐次実行単位及び排他的逐次実行単位のうち特定制御指示入力手段によって指定された逐次実行単位の実行を停止または再開するように動作するので、逐次実行単位ごとの詳細な制御指示を行うことにより、他の排他的逐次実行単位の実行を停止させる間に微妙なタイミングの設定をすることが可能となる。

【0293】請求項12及び36記載の本発明では、請求項9記載の発明の作用に加えて、逐次実行単位の状態指標をデバッグ制御に反映させるので、デバッグ対象のプログラムを、デバッグ装置を介さないで実行する際には起こり得ないタイミングの実行が回避され、さらに効率的なデバッグが可能となる。

【0294】請求項13及び37記載の本発明では、制御指示入力手段に特定の逐次実行単位を対象として第1の単位実行指示が入力されたときに次に実行すべき命令が内在的停止命令である場合には特定の逐次実行単位を実行停止状態とし、該制御指示入力手段に第2の単位実行指示が入力されると次の命令を実行可能な状態にするように制御するので、第1の単位実行指示と第2の単位実行指示との間に他の排他的逐次実行単位を実行するこ

10

20

30

40

50

とにより微妙なタイミングの設定をすることが可能となる。

【0295】請求項14及び38記載の本発明では、プログラムに含まれる各モジュールを、着目属性に応じて分類し、特定の着目属性を持つモジュールに含まれる命令群総てをプログラムの実行の際に同一に扱うように制御するので、該プログラムに含まれるモジュールをその被着目度に応じて適切に取り扱うことが可能となる。

【0296】請求項15及び39記載の本発明では、請求項14記載の発明の作用に加えて、分類手段が、プログラムに含まれる各モジュールに対する分類規則を用いて前記各モジュールを分類する手段と、変更指示に基づいてその分類を更新する手段とを具備するので、そのモジュールの属性をデバッグ作業者が変更することが可能となり、より効率的なデバッグが可能となる。

【0297】請求項16及び40記載の本発明では、着目しないモジュールで定義された命令群、例えば関数、メソッド、手続き等の中に入り込むような『STEP IN』命令は、当該命令群全体をひとかたまりに実行してしまう『STEP OVER』命令と解釈するので、着目の対象でないモジュールに対する容易な実行制御が可能となる。

【0298】請求項17及び41記載の本発明では、プログラムに含まれる各モジュールを分類し、このモジュールの分類を考慮して逐次実行単位を分類し、この各逐次実行単位の分類に適応させてデバッグ制御を行うので、請求項1及び2に係る本願発明の目的とするところ及び請求項15に係る本願発明の目的とするところを同時に達成すること、即ち、着目の対象でないモジュールに対する容易な実行制御が可能となると同時に、モジュールの着目属性を逐次実行単位の分類に反映させた上で逐次実行単位間の実行のタイミングを任意に定めることが可能となる。

【0299】請求項18及び42記載の本発明では、デバッグログ手段に記録されたデバッグ情報を用いることにより、デバッグ装置上でデバッグ実行された場合の各逐次実行単位間の動作のタイミングと同じ、又は異なるタイミングで該各逐次実行単位を動作させる動作過程を生成するので、例えばデバッグ作業者に提示することでプログラムに含まれる問題発見が容易になる。

【0300】請求項19及び43記載の本発明では、請求項18記載の発明の作用に加えてさらに、請求項18記載の発明によって生成された動作過程を自動実行し、その結果とデバッグ情報とを比較する手段を備えるので、効率的な自動テストが可能となる。

【0301】請求項20及び44記載の本発明では、デバッグ対象のプログラムに含まれる並列して実行される複数の逐次実行単位を属性に応じて分類し、該逐次実行単位ごとに設けられたウインドウに特定の属性を持つ逐次実行単位の動作状態を表示し、該ウインドウごとに、

制御指示が入力されるように構成するので、並列プログラムのデバッグの動作状態を並列的に可視化することが可能となり、より効率的なデバッグが可能となる。

【0302】請求項21及び45記載の本発明では、請求項20記載の発明の作用に加えて、デバッグ対象のプログラム実行中の逐次実行単位の増減及び逐次実行単位の分類の変化に対応させてウインドウが増減されるので、逐次実行単位の状態をより反映させたデバッグを行うことができ、より効率的なデバッグが可能となる。

【0303】請求項22及び46記載の本発明では、デバッグ対象のプログラム全体に対する第1の制御指示を入力するための手段と、デバッグ対象のプログラムに含まれる並列して実行される複数の逐次実行単位を分類し、該逐次実行単位ごとに設けられたウインドウに、特定の属性の対応する逐次実行単位の動作状態と、該ウインドウに対応する逐次実行単位のみを対象とする第2の制御指示を入力するための手段とを併有させることにより、より効率的なデバッグが可能となる。

【0304】請求項23及び47記載の本発明では、逐次実行単位ごとのウインドウ画面内で逐次実行単位の実行状況に適応させた前記第2の制御指示を入力する手段のための表示を行うので、効率のよいデバッグが可能となる。

【0305】請求項24及び48記載の本発明では、デバッグ対象のプログラムに含まれる並列して実行される複数の逐次実行単位のうち特定の逐次実行単位ごとに可視的に設けられたウインドウごとに、該特定の逐次実行単位が実行中のモジュールもしくは命令群を表示させ、更に、そのウインドウに前記関数を呼出す論理軌跡を表示するので、並列プログラムのデバッグの動作状態を並列的に可視化し、デバッグ対象のプログラムの働きを理解する助けとなる。

【0306】以上から、本発明の構成によれば、例えば複数のスレッドの実行を、デバッグ作業者の希望する順序で実行制御することが可能となり、従ってデバッグ作業者は、タイミングに起因するバグの再現を常に行うことができるようになる。

【0307】従って、並列プログラム、例えばマルチスレッドプログラムをデバッグするユーザは、スレッド間の実行を任意の順序でステップ実行することが可能となり、これによって、従来のデバッグ装置では再現することが困難であったプログラム中の誤りを非常に効率良くデバッグすることができるようになる。

【図面の簡単な説明】

【図1】本発明の一実施形態に係り、デバッグ装置の用いられる環境を示す概念図である。

【図2】本発明の一実施形態に係り、マルチスレッドプログラムのデバッグ装置の構成を示す概念図である。

【図3】本発明の一実施形態に係り、マルチスレッドプログラムのデバッグ装置の構成を、デバッグ作業及び

51

ターゲットプログラムとの関係において示した概念図である。

【図 4】本発明の一実施形態に係り、デバッグコマンドの種別を表した図である。

【図 5】本発明の一実施形態に係り、スレッド分類部及びスレッド表示部の働きを説明するための概念図である。

【図 6】本発明の一実施形態に係り、一般／スレッド別デバッグコマンドを与えられた排他実行／同時実行／継続実行スレッド別の振る舞いの対応関係を示す概念図である。

【図 7】本発明の一実施形態に係り、デバッグ装置の動作を示すフローチャートである。

【図 8】本発明の一実施形態に係り、デバッグ装置の動作を示すフローチャートである。

【図 9】本発明の一実施形態に係り、デバッグ装置の動作を示すフローチャートである。

【図 10】本発明の一実施形態に係り、デバッグ装置の動作を示すフローチャートである。

【図 11】本発明の一実施形態に係り、デバッグ装置の動作を示すフローチャートである。

【図 12】本発明の一実施形態に係るデバッグ装置の画面を表す概念図である。

【図 13】本発明の一実施形態に係り、一般コントロールパネルを示す概念図である。

【図 14】本発明の一実施形態に係るデバッグ装置の画面を表す概念図である。

【図 15】本発明の一実施形態に係るデバッグ装置の画面を表す概念図である。

【図 16】本発明の一実施形態に係るデバッグ装置の画面を表す概念図である。

52

【図 17】本発明の一実施形態に係り、マルチスレッドプログラムのデバッグ装置の構成を示す概念図である。

【図 18】本発明の一実施形態に係り、関数分類に関して説明するための概念図である。

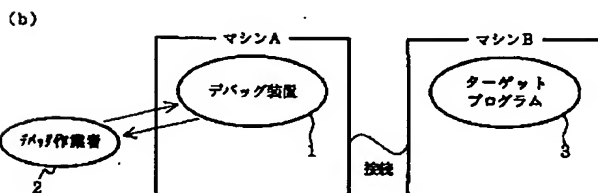
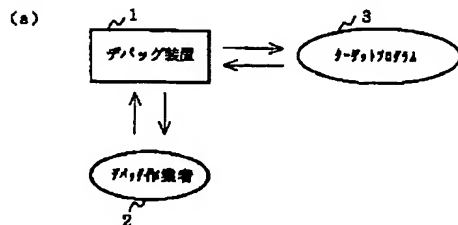
【図 19】本発明の一実施形態に係り、マルチスレッドプログラムのテスト装置の構成を示す概念図である。

【図 20】従来のデバッグ装置の画面を概念的に表した図である。

【符号の説明】

- 1 デバッグ装置
- 101 CPU
- 107 記憶装置
- 109 ターゲットプログラム読み込み部
- 111 ターゲットプログラム実行部
- 113 ターゲットプログラム停止部
- 115 排他的スレッド実行機構
- 117 スケジューラ部
- 121 スレッド分類部
- 123 スレッド分類規則格納部
- 125 デバッグコマンド表示部
- 127 デバッグコマンド入力部
- 129 スレッド表示部
- 131 スレッド表示規則格納部
- 1201 スレッドトレーススインドウ
- 1202 スレッドコントロール部
- 1404 スタックフレーム選択表示部
- 1405 スレッド別デバッグコマンド指示部
- 1701 関数分類部
- 1703 関数分類規則格納部
- 1901 テストパターン生成部
- 1903 デバッグログ記録部

【図 1】



【図 4】

デバッグコマンド

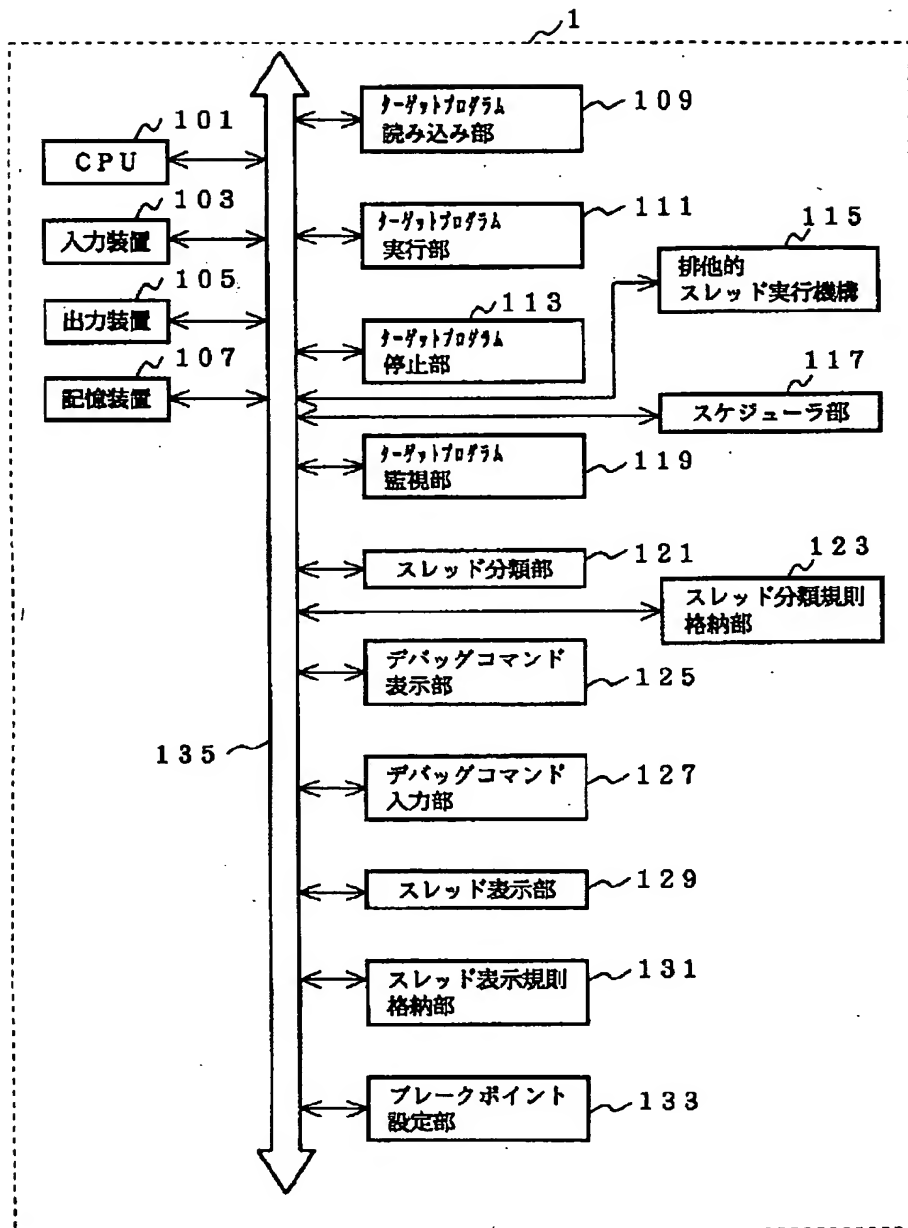
一般デバッグコマンド

- ターゲットプログラムの指定、読み込み
- デバッグの開始
- デバッグの終了
- 全停止スレッドの実行再開
- 全実行中スレッドの停止
- ブレークポイント設定
- ターゲットプログラム状態（グローバル変数値等）の表示
- ...

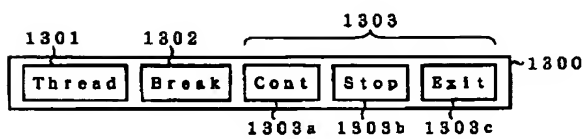
スレッド別デバッグコマンド

- 停止スレッドの実行開始
- 実行中スレッドの停止
- Step In
- Step Over
- Step Out
- スレッド別ブレークポイントの設定
- ローカル変数（現在のスタックフレーム、ローカル変数等）の表示
- 表示するスタックフレームの上げ下げ
- ...

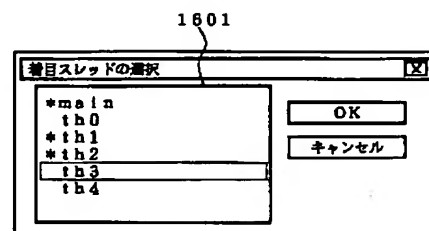
【図 2】



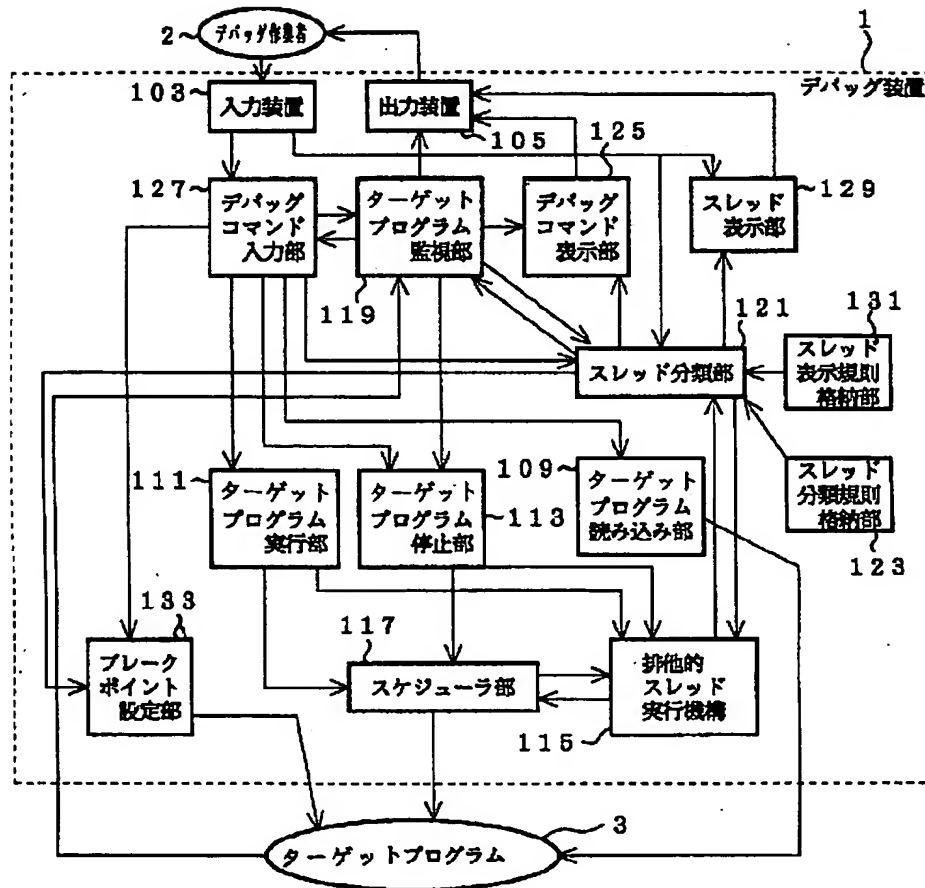
【図 13】



【図 16】



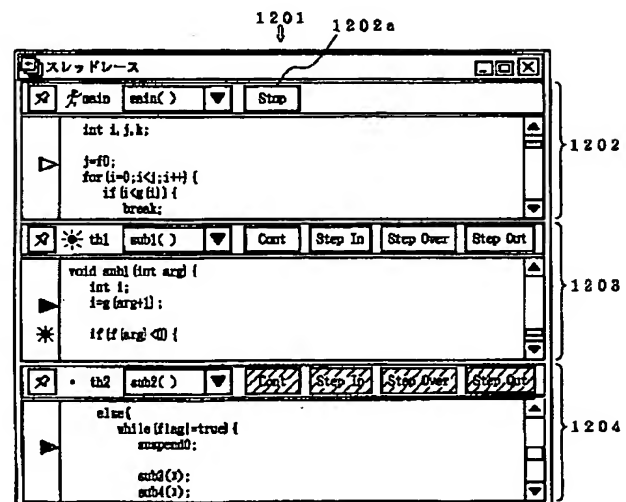
【図 3】



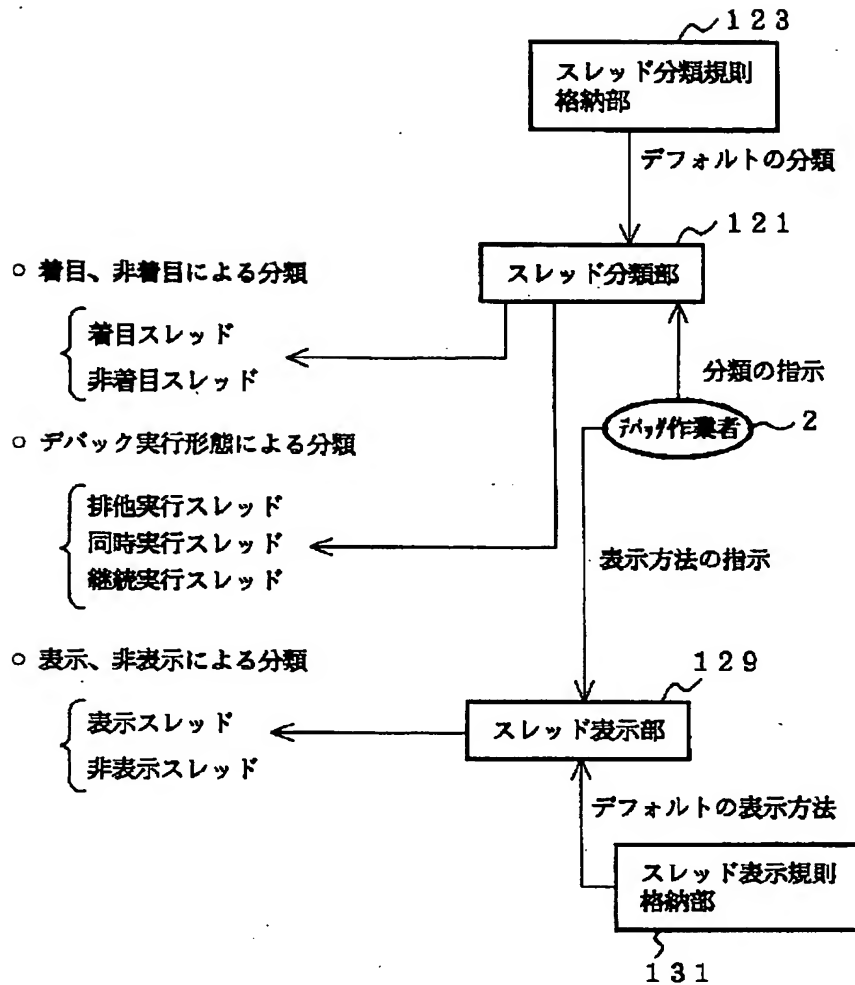
【図 6】

		排他実行 スレッド	同時実行 スレッド	継続実行 スレッド
一般デバッグ コマンド	実行	続けて再開	続けて再開	現在の実行状態 を維持
	停止	続けて停止	続けて停止	現在の実行状態 を維持
スレッド別 デバッグ コマンド	実行	対象スレッド を再開	続けて再開	現在の実行状態 を維持
	停止	対象スレッド を停止	続けて停止	現在の実行状態 を維持

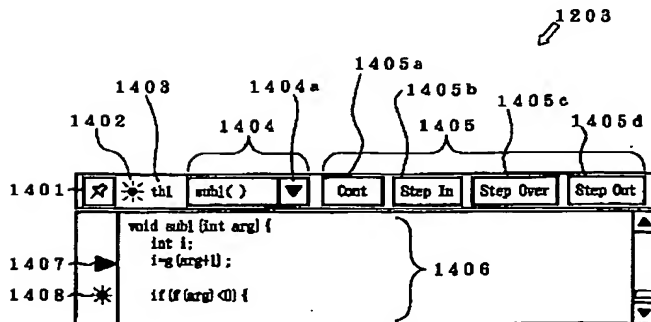
【図 12】



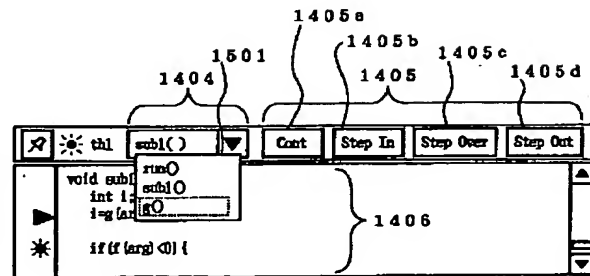
【図5】



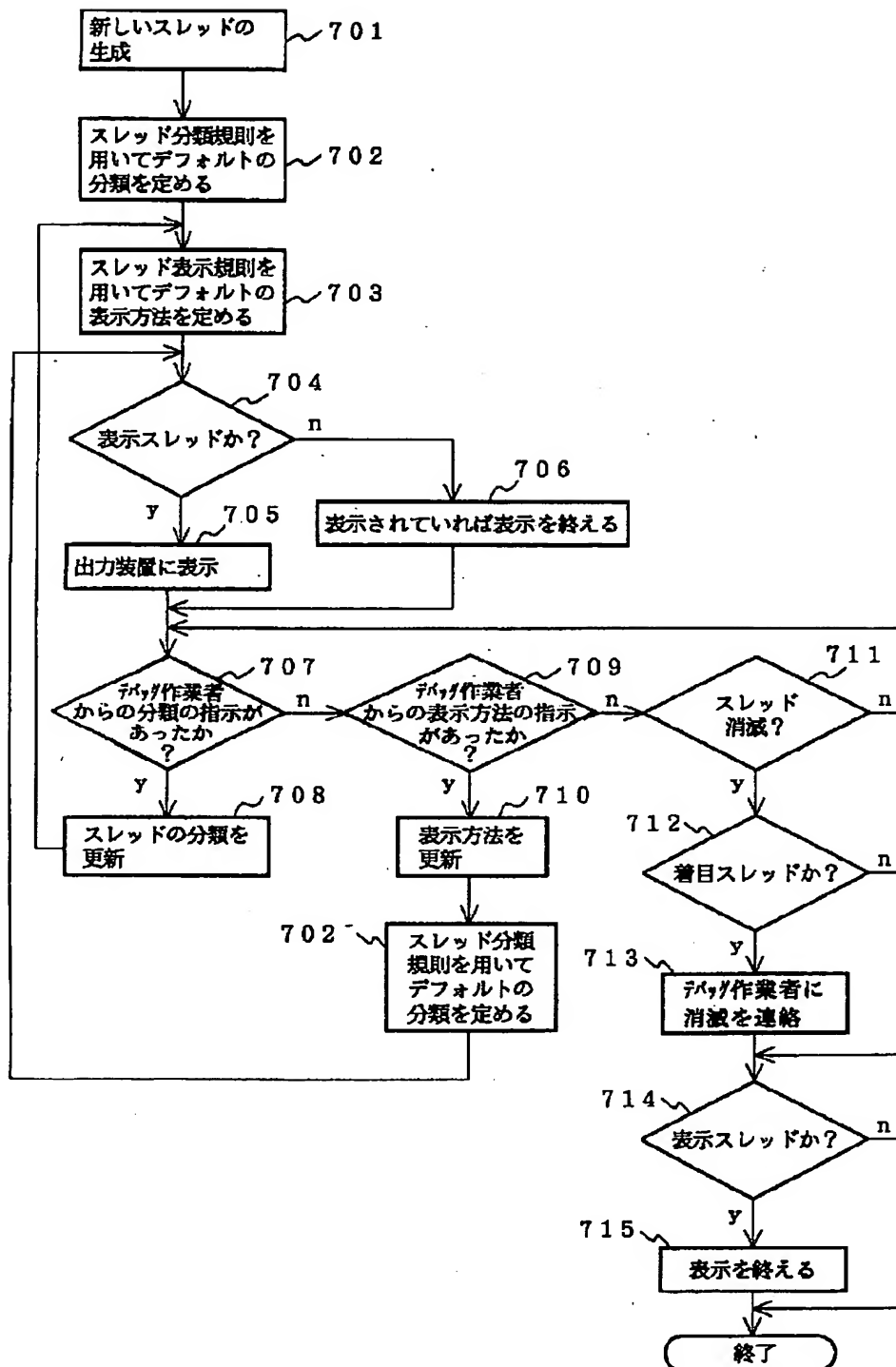
【図14】



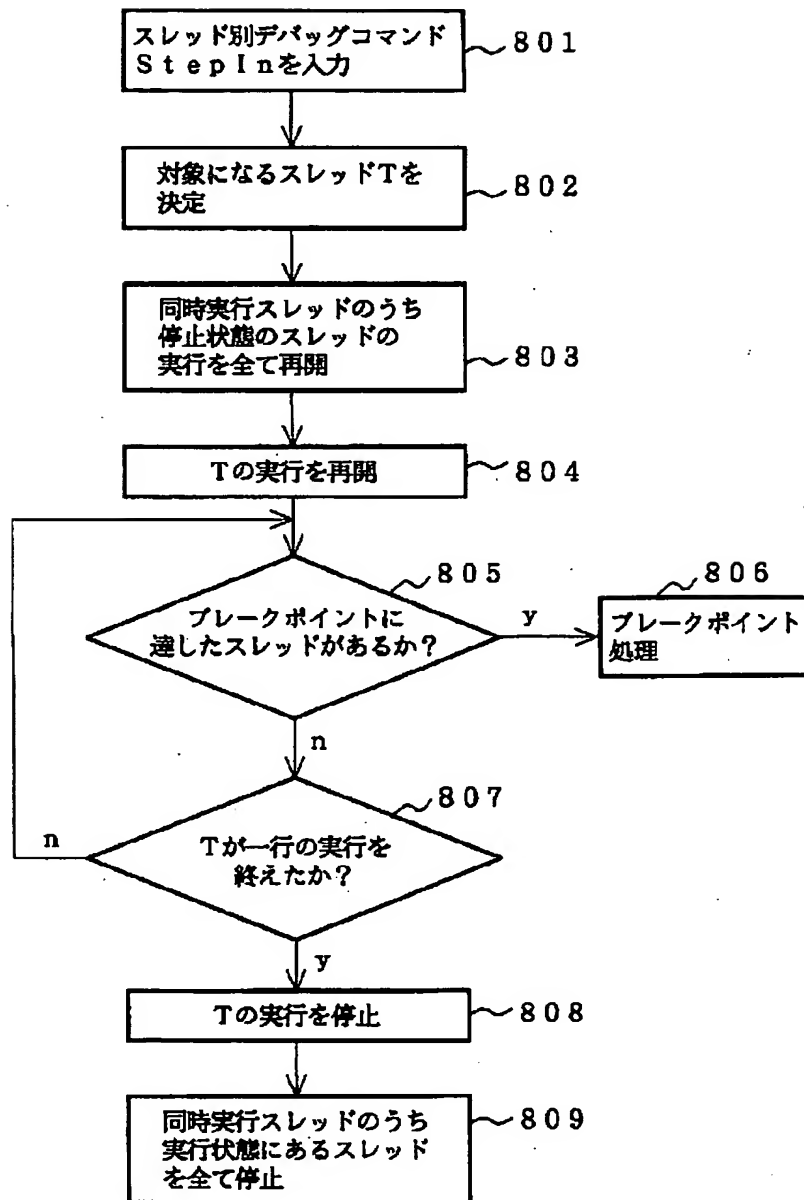
【図15】



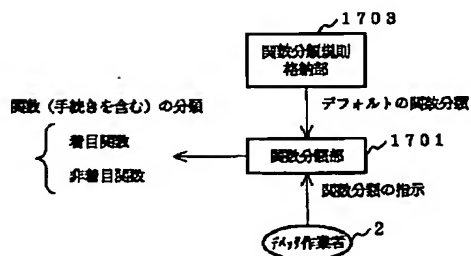
【図 7】



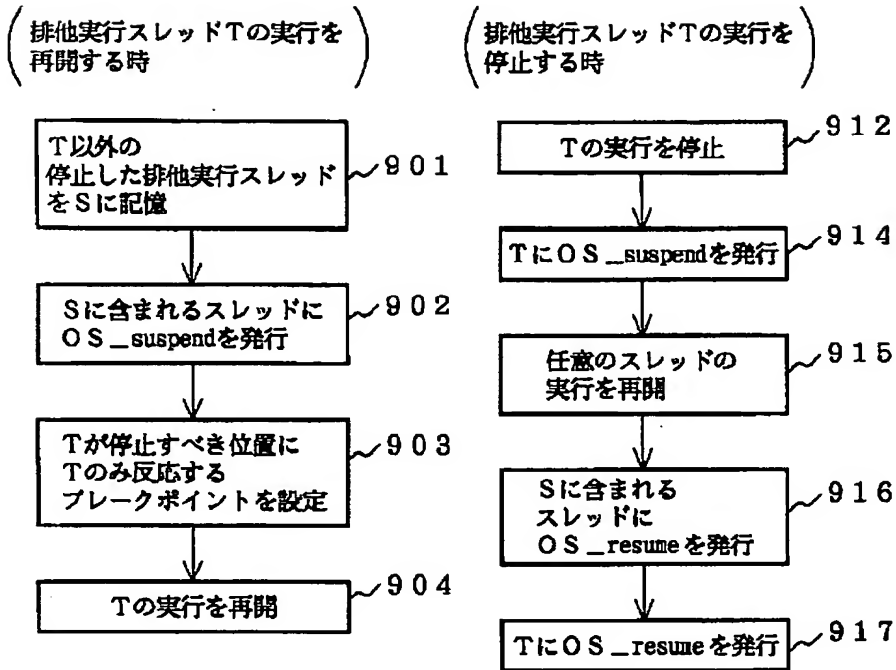
【図 8】



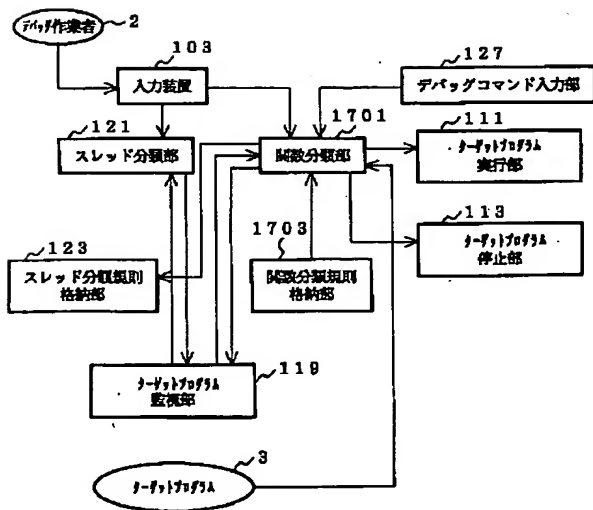
【図 1 8】



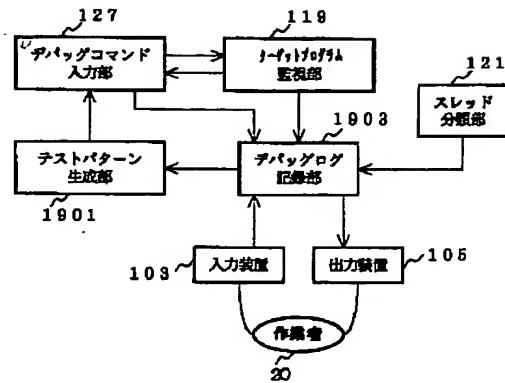
【図 9】



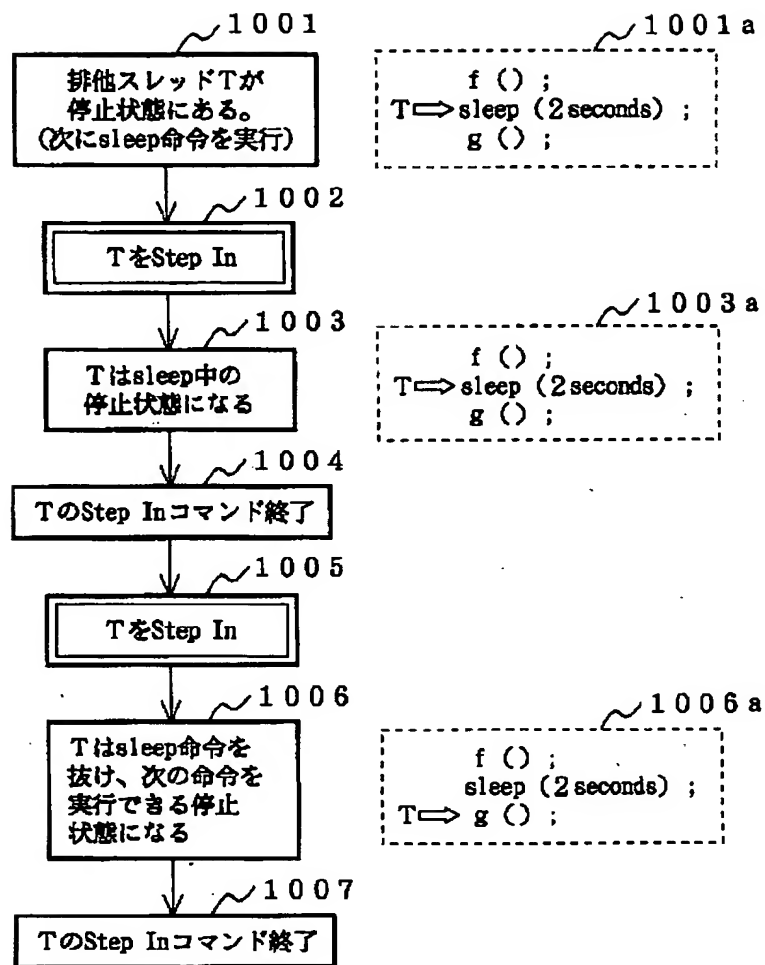
【図 17】

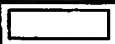


【図 19】

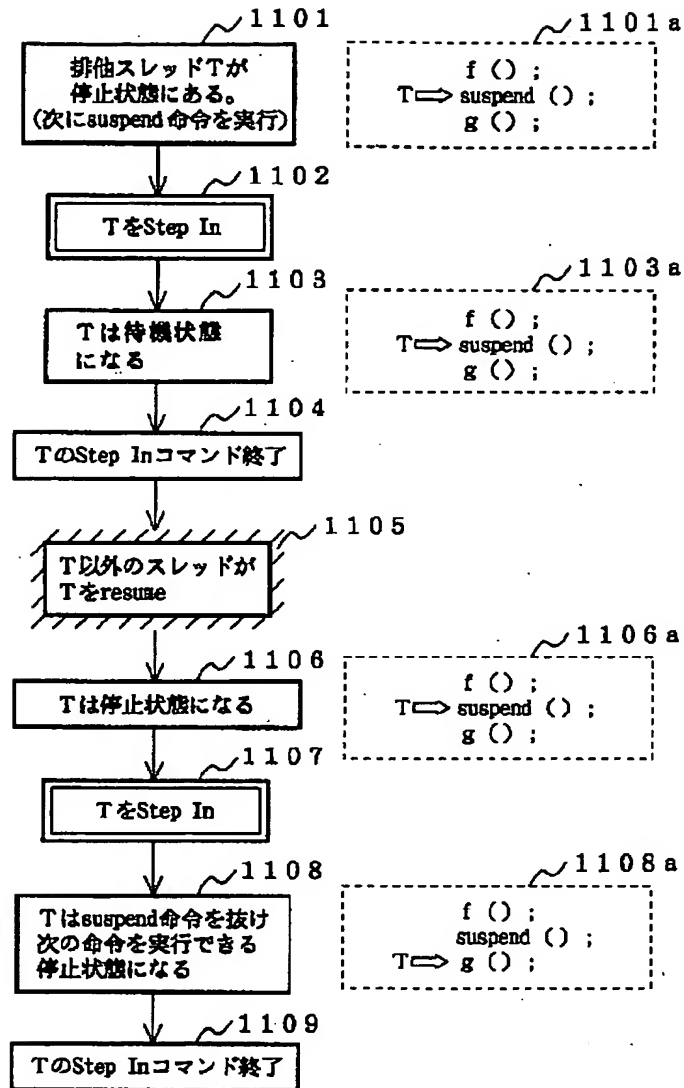


【図 1 0】

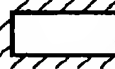


( はデバッグ作業者による
デバッグコマンド)

【図 1 1】



はデバッグ作業による
デバッグコマンド



はあるスレッドによる
ターゲットプログラムの実行

【図 20】

